RCA

RCA COSMAC VIP
Instruction Manual for VP-111

VIP-330

Suggested Price $5.00

RCA COSMAC VIP

Instruction Manual for VP-111

RCA COSMAC VIP MARKETING
New Holland Avenue
Lancaster, PA 17604

Contents

Printed in USA/1-80

I. Getting Started

COSMAC VIP (Video Interface Processor) VP-111 is a complete computer on a single printed-circuit card. It includes the following:

- RCA CDP1802 Microprocessor (91 instructions)
- 1024-byte RAM
- Built-in hex keyboard (reliable touch-pad type)
- Graphic video display interface (standard video output)
- 100-byte-per-second audio cassette interface
- Crystal clock
- 512-byte ROM operating system
- Documentation
- Ready-to-use video game programs
- Unique CHIP-8 Language (31 easy-to-use instructions)

Plus this expansion capability:

- On-card RAM expansion up to 4096 bytes
- On-card parallel I/O port
- Connector for extensive external expansion capability
- VIP expansion accessories including:
  VP-590 Color Board
  VP-595 Simple Sound Board
  VP-550 Super Sound
  VP-570 4K RAM Expansion Board
  VP-560 EPROM Board
  VP-565 EPROM Programmer Board
  VP-580 Expansion Keyboard
  VP-700 Tiny BASIC ROM Board and many more. See the complete list and brief description at the end of this chapter.

What You Will Need:

To get started with your VIP it is necessary to install the cables, cable ties, cable tags and rubber feet. This chapter contains complete instructions for doing so.

To operate the VP-111 you will need the following accessories:

1. A video display — either a video monitor (15,750 horizontal sync, 60Hz) or an inexpensive TV set and an approved RF modulator.
2. A 5-volt power supply (see further information in this chapter) with a current capability of at least 250ma. As you add accessories, particularly memory, the current requirement will increase.
3. An inexpensive standard cassette tape recorder for data storage. Panasonic RQ-2309 or equivalent recommended.
4. Any 8-ohm speaker.

This chapter will instruct the user in wiring in the power supply and speaker.

Assembly:

The VP-111 comes assembled, with the exception of the cables, cable ties, cable tags and feet. The VP-111 has been completely tested.

Caution:

To insure the inherent high reliability of the VP-111, which utilized CMOS integrated circuits, avoid situations where static electricity may discharge to your VIP.

STATIC DISCHARGE from finger contact or ungrounded equipment such as tools and soldering irons MAY CAUSE PERMANENT DAMAGE TO THIS PRODUCT!

Rugs, coats, sweaters and shoes especially those made of synthetic materials such as nylon, may cause substantial static build-up on persons

who contact such materials. In most areas this problem is more severe during the winter months.

To Protect Your VIP . . .

- Avoid situations which may cause static build-up. Antistatic spray preparations, available in supermarkets and retail stores, can be applied to rugs and other synthetic material which could develop static.
- Use only properly grounded soldering irons and tools.
- Touch a grounded object immediately before handling the VIP PC board or accessory PC boards.

Preparing the Cables:

If the cables are not prestripped, prepare the cables as shown in Figure 1. The video cable is the larger diameter of the three cables. The audio cables for tape-in or tape-out are identical.

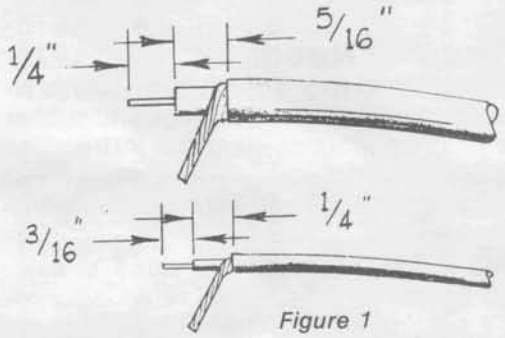


Figure 1

Installing the Cables:

Using a low temperature small tipped, grounded soldering iron and a small gauge rosin core solder, solder the cables in place. Use Figure 2 as a guide. "TAPE IN", TAPE OUT" "VIDEO" and "GND" are labeled on the circuit board.

Attach the appropriate cable tag to each cable by removing the protective paper backing and folding the tag around the cable.

Again using Figure 2 as a guide, install cable ties to provide strain relief. Do so by threading cable tie through the appropriate PC board holes. Insert cable tie tail through the cable tie head with the cable captured in the loop. Cinch tight and trim off excess tail.

About the Power Supply & Hooking It Up:

The VP-111 may be powered in two ways — a regulated 5 ± .25 volt power supply or an unregulated 8 to 10 volt power supply with a 7805 regulator (not provided) and heat sink (not provided) mounted on board. Either must have a current capability of at least 250ma.

To install a regulated power supply, connect the positive lead to the large hole in the upper left-hand corner marked "+VDC." Refer to Figure 3. Solder the ground wire to the hole marked "GND."

CAUTION: *Reversing the polarity of these wires will cause permanent damage to the VP-111.* Install cable tie strain relief as illustrated.

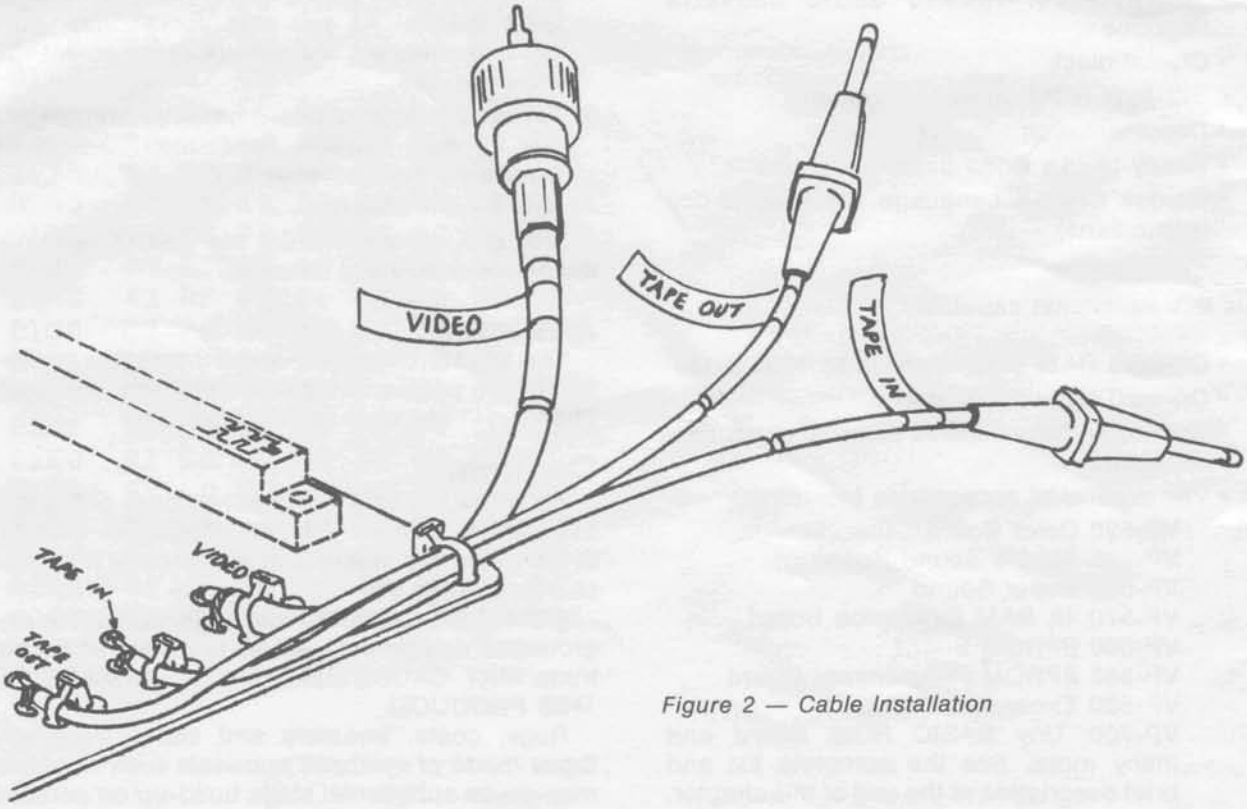


Figure 2 — Cable Installation

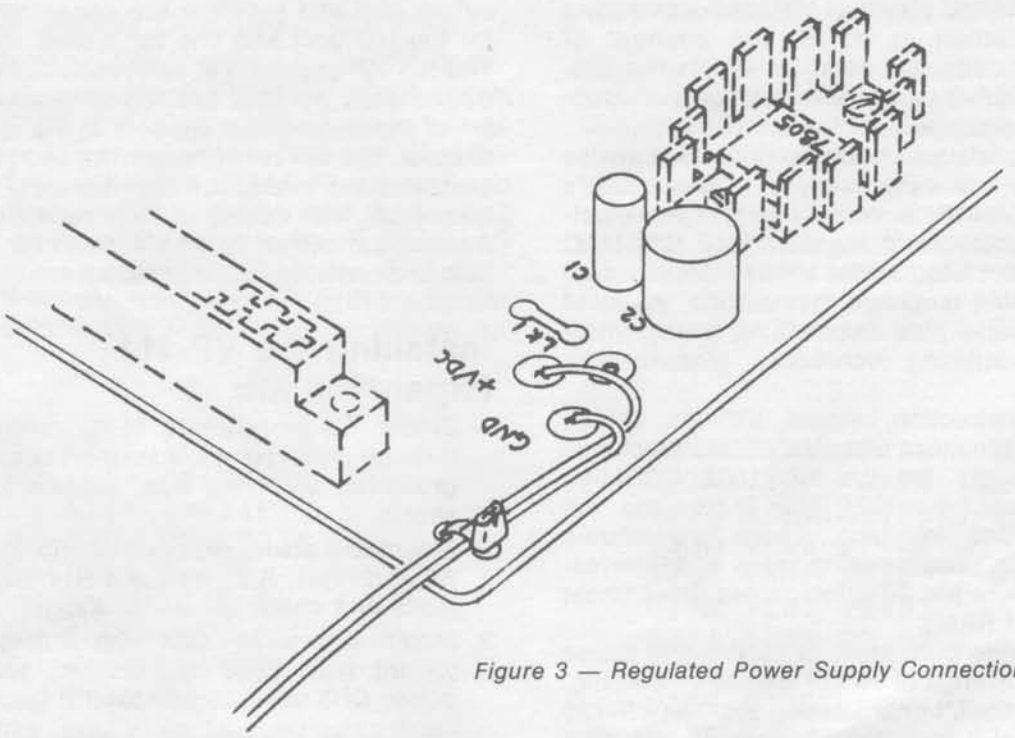


Figure 3 — Regulated Power Supply Connection

To use an unregulated power supply Link 1, marked "LK1" on PC board, *must be cut.* Mount U28, a 7805 5-volt positive regulator IC (not supplied), to a heat sink (not supplied) and solder regulator leads in place. See Figure 3. While only 250ma is required for the VP-111 as is, this configuration can supply up to 1 amp to an expanded VP-111.

Solder unregulated input power leads to the circuit board, at the location marked "+VDC" and "GND" in the upper left corner. *Be sure polarity is correct.*

The Speaker:

Solder the leads of any 8-ohm speaker to the terminals marked "SPKR" on the left side of the PC board.

A Footnote:

The last step in completing the VP-111 is to install the rubber feet. Press a self-adhesive foot on the large dot in each corner on the reverse side of the PC board and one on the dot immediately under the keypad.

Covering It Up:

An attractive, plastic dust cover, VP-55, is available for the VP-111. It will protect the components and minimize the possibility of shorts which could permanently damage the VP-111. The cover can be used on the standard VP-111 or expanded VP-111. The VP-111 is fully operational with the cover in place. The cover *should not* be used if more than .5 amps is required by the system when a 7805 regulator is on board in combination with an external unregulated power supply.

Turning It On:

After attaching the video cable to a video display (monitor or modulator and TV set) and positioning the switch at reset, "RES," apply power. The power LED should go on. If not check the power supply polarity and voltage.

Hold hex key C down while you position the run switch at "RUN." You should hear a tone with key C pressed and the Q light should be on. When you release key C the tone and Q light should both go off. The tone occurs whenever the Q light is on. You should now see a random pattern of small square spots on the display. Push hex keys 8008 in sequence and you should see 8008 at the bottom left of the screen and 64 at the lower right. Adjust your display controls for the best picture (white spots on a black background). A random moving pattern of spots in the lower right of the display is normal. This is the operating system scanning the keyboard for a key press.

After completing the above set-up procedure, you are ready to enter and run programs on your COSMAC VIP.

Chapter 2 covers the operation of the VIP, reading from and writing to memory or cassette tape. Chapter 3 covers the impressive CHIP-8 interpretive language and Chapter 4 covers 1802 machine language. Chapters 5, 6 and 7 include the operating system listing, logic diagrams and video game listings.

Have fun!

More Information?:

If you are a beginner, the VIP User's Guide, VIP-320, is highly recommended. This manual covers

the getting started phase of VIP microcomputing — VIP operation in detail, the concept of variables, hexidecimal notation, displaying patterns, the CHIP-8 interpretive language instruction set with examples and more.

For the experienced hobbyist or serious novice and for general reference, the 1802 User's Manual, MPM201B, is an excellent, comprehensive documentation of the CDP1802 COSMAC microprocessor used by the VP-111. This manual covers machine language instructions, all 91 of them, in detail — plus architecture, timing, interfacing, programming techniques, glossary and much more.

The VIP Instruction Manual, VIP-311, covers VIP operation in more technical detail. It includes the data sheets for the CDP1802 COSMAC microprocessor, the CDP1832 ROM, and the CDP1861 video interface. There are trouble shooting hints, programming ideas and interfacing ideas. There are 20 video games listed (most require 2K of RAM).

For the game buff, there is the VP-710 Game Manual with listings of 16 VIP games — Bowling, Biorythm, Pinball, Lunar Lander, etc. The VP-720 Game Manual II includes 16 more — including some of the best VIP games we've seen — Backgammon, Two Player Blackjack, Miniature Golf, etc. (2 to 4K of memory required).

These manuals can be ordered from your local dealer or from RCA, VIP Customer Service, New Holland Avenue, Lancaster, PA 17604.

Expanding the VP-111:

The VP-114 expansion kit will fully expand the on-board memory capability of the VP-111 to 4096 bytes. It adds an 8 bit input port and an 8 bit output port and provides two connectors — one for the I/O port and one for system expansion. This full VIP capability allows you to use any of the educational, low cost, exciting VIP accessories. A list of these products appears at the end of this chapter. The VP-114 expansion kit and the VIP accessories are available from your local dealer or from RCA, VIP Customer Service, New Holland Avenue, Lancaster, PA 17604. Send for complete data and ordering information.

Installing the VP-114 Expansion Kit:

1. Solder 18 pin sockets in IC positions U18 through U23. Use a low temperature, small tip grounded soldering iron. Inspect for solder shorts.
2. Mount and solder resistors R3, R5, R6, R7, R8, R9, R10, R11, R12, R13, and R14. Clip excess leads and check for solder shorts.
3. Mount and solder CR1 with cathode (band) toward right edge of PC card. Mount and solder CR3 with cathode toward back edge of card.
4. Mount and solder U24, U25, U26, and U27. Observe COS/MOS handling precautions. If you prefer to use sockets you will need two 16-pin sockets and two 24-pin sockets. Inspect for solder shorts and double-check insertion direction.
5. Observing insertion direction, insert 2114 RAM IC's into sockets, U18 through U23. Insertion direction is the same as the direction of the factory soldered in RAM IC's, U16 and U17. Inspect for bent pins.

VIP ACCESSORIES AND PRODUCTS

RCA COSMAC VIP PRODUCTS

Type	Description
VP-111	**Microcomputer** Assembled* & tested microcomputer. Cassette & video interface, 1K RAM, 512-byte ROM operating system, hex keypad, manual including chips & 5 video games. *User need only connect cables (provided) and a 5-volt power supply & speaker.
VP-114	**VP-111 Expansion Kit** Expands VP-111 to full VIP capability. Includes 3K RAM and sockets, components for 8 bit input & 8 bit output port. Sockets for system expansion & I/O port.
VP-711 (CDP18S711)	**VIP - Hobby Computer** An assembled microcomputer. Built-in cassette interface, video interface, 16-key keypad, 2K RAM, ROM operating system, CHIP-8 language and power supply. Output drives video monitor or rf modulator.
VP-44 VP-45	**RAM On-Board Expansion Kits** Four type 2114 (VP-44) or type 9131 (VP-45) RAM IC's for expanding the VIP on-board memory to 4K bytes. (Early kits used 9131 IC's — All others use 2114 IC's.)
VP-590	**VIP Color Board** Displays VIP output in color! Program control of four background colors and eight foreground colors. CHIP-8X language adds color commands. Includes two sockets for VP-580 Expansion Keyboards.
VP-595	**VIP Simple Sound Board** Provides 256 different frequencies in place of VIP single-tone output. Ideal for use with VP-590 Color Board for simultaneous color and sound. Great for simple music or sound effects! Includes speaker.
VP-550	**VIP Super Sound Board** Turn your VIP into a music synthesizer! Provides two independent sound channels. Frequency, duration and amplitude envelope (voice) of each channel under program control. On-board tempo control. Provision for multi-track recording or slaving VIP's. Output drives audio preamp. Does not permit simultaneous video display.
VP-551	**VIP Super Sound 4-Channel Expander Package** VP-551 provides four (4) independent sound channels with frequency duration and amplitude envelope for each channel. Package includes modified VP-550 super sound board, VP-576 two board expander, data cassette with 4-channel PIN-8 program, and instruction manual. Requires 4K RAM system and your VP-550 Super Sound Board.
VP-570	**VIP Memory Expansion Board** Plug-in 4K static RAM memory. Jumper locates RAM in any 4K block in first 32K of VIP memory space.
VP-580	**VIP Expansion Keyboard** Adds two-player interactive game capability. 16-key keypad with cable. Connects to sockets provided on VP-590 Color Board or VP-585 Keyboard Interface Board.
VP-585	**VIP Keyboard Interface Board** Interfaces two VP-580 Expansion Keyboards directly to the VIP. Not required when VP-590 Color Board is used.
VP-560	**VIP EPROM Board** Interfaces two Intel 2716 EPROMs to VIP. Places EPROMs anywhere in VIP memory space. Can also re-allocate on-board RAM in memory space.
VP-565	**VIP EPROM Programmer Board** Programs Intel 2716 EPROMs with VIP. Complete with software to program, copy, and verify. On-board generation of all programming voltages.

Type	Description
VP-575	**VIP Expansion Board** Plug-in board with 4 buffered and one unbuffered socket. Permits use of up to 5 Accessory Boards in VIP Expansion Socket.
VP-576	**VIP Two-Board Expander** Plug-in board for VIP I/O or Expansion Socket permits use of two Accessory Boards in either location.
VP-601	**ASCII Keyboard** Fully encoded, 128-character ASCII alphanumeric keyboard. 58 light touch keys (2 user defined). Selectable. "Upper-Case-Only".
VP-611	**ASCII/Numeric Keyboard** ASCII Keyboard identical to VP-601 plus 16 key numeric entry keyboard for easier entry of numbers.
VP-620	**Cable: ASCII Keyboards to VP-711** Flat ribbon cable, 24" length, for connecting VP-601 or VP-611 and VP-711. Includes matching connector on both ends.
VP-623	**Cable: ASCII Keyboards** Flat ribbon cable, 36" length with mating connector for VP-601 or VP-611 Keyboards. Other end is unterminated.
TC1210	**9" Video Monitor** Ideal, low-cost monochrome monitor for displaying the video output from your VIP or other computers.
TC1212	**12" Video Monitor** A large (74 sq. in. picture) monochrome monitor for use with your VIP or other computers with video output.
TC1217	**17" Video Monitor** A really BIG monochrome monitor for use with your VIP or other computers with video output. 148 sq. in. pictures.
CDP18S731 CDP18S745	**RAM/IO Expansion Kits for Older 18S022 Kits** Four type 9131 (18S731) or 2114 (18S745) RAM IC's plus other components for I/O expansion ports.
VP-700	**VIP Tiny BASIC ROM Board** Run Tiny BASIC on your VIP! All BASIC code stored in ROM. Requires separate ASCII keyboard.
VP-710	**VIP Game Manual** More exciting games for your VIP! Includes Blackjack, Biorythm, Pinball, Bowling and 20 others.
VIP-311	**VIP Instruction Manual** (Included with VP-711)
VIP-320	**VIP User Guide Manual** (Included with VP-711)
MPM-201B	**CDP1802 User Manual**
VIP-330	**VP-111 Instruction Manual** (Included with VP-111)
VP-55	**Plastic Cover for VP-111**
VP-720	**VIP Game Manual II** More exciting VIP games including some of the best we've seen. Miniature Golf, Backgammon, Two Player Blackjack, Knockout in color ... 16 in all. Available 2Q '80.

FOR PRICE LIST CONTACT:
RCA COSMAC VIP Marketing
New Holland Avenue, Lancaster, PA 17604
Phone: (717) 397-7661

II. VP-111 Operation

COSMAC VIP is operated with the RUN switch and hex keyboard. The PWR light shows that power is on. The Q light is activated by various programs. A tone is sounded whenever the Q light is on. The TAPE light glows when cassette input data is present. When using COSMAC VIP, always start with the RUN switch in the down (or reset) position. Flipping the RUN switch up initiates execution of machine language programs beginning at memory location 0000. If you have previously stored the CHIP-8 language interpreter program at locations 0000-01FF, execution of a program written in this language will begin at 0200. To manually terminate execution of any program, flip RUN down.

Using the Operating System

With COSMAC VIP you can load programs into memory from the hex keyboard or cassette recorder, record the contents of memory on cassettes, show the contents of memory bytes in hex form on the display, and examine the contents of CDP1802 microprocessor registers. These functions are performed with the aid of a special program called an operating system. This operating system is contained in a ROM so that it's ready to use as soon as power is turned on. It is located at memory locations 8000-81FF. A machine code listing and summary of this operating system is provided in Chapter 6.

To use the operating system hold key C down on the hex keyboard when you flip RUN up. You will hear a tone. Release key C and you're ready to use the operating system.

After selecting the operating system you can do four different operations as shown in the following table:

KEY	OPERATION
0	MW (Memory Write)
A	MR (Memory Read)
F	TW (Tape Write)
B	TR (Tape Read)

For any of these operations you must first enter a memory address. Enter the 4 hex digits of any memory address using the hex keyboard (most significant digit first). You will see the address at the lower left of the screen and the byte contained in that address at the lower right. Remember that addresses and bytes are always entered and shown in hex form. Suppose you entered 0200. You will see 0200 at the bottom left of the screen and the byte stored at 0200 at the lower right.

Memory Write

If you want to change this byte, press the 0 key. Now press two digits of the new byte (most significant digit first) and it will be stored at 0200 replacing the original byte. You will see this change on the screen. If you enter another byte it will be shown and stored at the next higher address in sequence (0201 in this example). You can load any sequence of bytes directly from the hex keyboard in this manner. If you make a mistake, flip RUN down. With key C pressed, flip RUN back up. Enter the address at which you made the error. Press key 0 and resume entering your program.

Note the random bit pattern on the screen above the hex display. This pattern is the binary data contained in the last 256-byte page of the on-card RAM. If you have a 1024-byte RAM, you are seeing locations 0300-03FF on the screen. Bit 7 of the byte at 0300 is in the upper left corner. Try storing a sequence of eight AA bytes followed by eight 55 bytes starting at location 0300. Keep repeating this sequence to draw a checkerboard pattern on the screen. There are 32 rows of spots on the screen. Each row represents 8 memory bytes (64 bits). Locations 0300-0307 are shown in the top row, 0308-030F in the next row down.

Memory Read

Suppose you wish to examine the contents of a memory location. Flip RUN up while pressing key

C. Enter the address of the location you want to examine. Press key A for the Memory Read mode. You will see the memory address and the byte stored at that address on the screen. Press any hex key to step through memory and see the contents. Memory locations examined are left unchanged. If a program doesn't run properly you can use this mode to verify that it was stored correctly in memory.

You can now enter and run this short beeper program. Flip RUN up with key C pressed. Release key C and enter address 0000. Press key 0 to select the Memory Write mode. Now enter the beeper program one byte at a time using the hex keyboard. Flip RUN down to reset the computer. Flip RUN up to execute the beeper program you just loaded into locations 0000-000C. You can load and run any COSMAC VIP program in this manner. For most of the game programs you will first have to load the CHIP-8 interpreter into locations 0000-01FF followed by the game program starting at location 0200.

Beeper Program

This machine-language program flashes the Q light and beeps at a rate determined by the byte at location 0002. Change this byte for faster or slower rates.

```
0000   7A  F8  0F  BF  2F  9F  3A  04
0008   31  00  7B  30  01  00  00  00
```

Tape Write

Any program you load into memory will be lost when you turn off power. Unless it is safely stored, you will have to key it in by hand again the next time you want to use it. The cassette interface is provided so that after keying in a program you can then record it on an audio cassette; and when you want to use the program again, all you have to do is play it back into the memory from the cassette. This playback usually takes less than 30 seconds.

The COSMAC VIP cassette interface was designed to work with most standard audio cassette recorders. Panasonic models RQ-2309, RQ-2308, RQ-309DS, RQ-212D, and RQ-413S have yielded satisfactory results as has the Sony TC-150. In general, better quality recorders provide more reliable operation.

Your tape recorder must have an 8-ohm earphone or external speaker jack and a microphone input jack. Connect the "tape in" cable plug to the recorder's earphone or external speaker jack. Connect the "tape out" cable plug to the recorder's microphone input jack.

The memory is divided into 256-byte pages for recording. You can record 1 to 15 consecutive pages on tape. The low-order byte of your starting address should be 00. Select the operating system by holding key C down while flipping RUN up. Enter the 4-digit address of the first page to be recorded on tape. Press key F and you're ready to record. Rewind a blank cassette and place your cassette unit in the record mode. Wait about 10 seconds and tap the hex key that represents the number of pages you want to record on tape. The screen will go blank and you'll hear a tone while recording. When the specified number of pages has been recorded on the cassette, the tone will end and the last memory byte recorded on tape will be shown on the screen.

Tape Read

To load memory from a previously recorded cassette, first select the operating system (RUN and key C). Enter the memory address of the first page to be loaded (usually 0000). Press key B to select the Tape Read mode. Rewind and play the cassette. Immediately press the hex key representing the number of pages you want to load into memory from the cassette. The tape recorder tone control should be set to maximum high. The volume control should be set for a steadily glowing tape light when data is being read from the tape. The screen will go blank while the program is loaded from the tape into memory. It will show the last byte loaded into memory at the end of loading.

If the Q light and tone come on while a tape is being read, an error occurred. Flip RUN down, rewind the cassette, and try again. You may have to readjust the cassette volume control. Be sure that the cassette contains at least as many pages as you specify to be loaded. For most of the game programs, load the CHIP-8 interpreter program into 0000-01FF, then load the game program starting at 0200. Record a cassette from 0000 to the end of the game program. When you load this tape, starting at 0000, you will be ready to play the game.

Testing Your Cassette System

Test your cassette system by entering the beeper program at 0000. Store 25 at 00FF. Now record 1 page on a cassette starting at 0000. Load this 1 page back into memory from the cassette starting at 0000. If no errors occur you should see "OOFF 25" on the screen after loading is complete. Flip RUN down, then up, and the beeper program should be running.

After recording and checking a program cassette, you can break out the tabs at the top of the cassette to prevent accidental erasure. In the event you wish to record on a cassette after you have broken out the tabs, you can do so simply by pasting tape over the tab holes. You can record and keep your own cassette software library starting with the game programs in this manual. Cassette recording or playback should require 5 + 2.5N seconds. N is the number of pages recorded on tape. The next section describes how you can design your own programs using a unique easy-to-learn programming language called CHIP-8.

III. CHIP-8 Language Programming

CHIP-8 is an easy-to-learn programming language that lets you write your own programs. To use the CHIP-8 language, you must first store the 512-byte CHIP-8 language program at memory locations 0000 to 01FF. The CHIP-8 language program is listed in this chapter in hex form so you can enter it directly in memory using the hex keyboard. You can then record it on a memory cassette for future use. Each CHIP-8 instruction is a two-byte (4-hex-digit) code. There are 31, easy-to-use CHIP-8 instructions as shown in Table I.

When using CHIP-8 instructions your program must always begin at location 0200. There are 16 one-byte variables labeled 0-F. VX or VY refers to the value of one of these variables. A 63FF instruction sets variable 3 to the value FF (V3=FF). I is a memory pointer that can be used to specify any location in RAM. An A232 instruction would set I=0232. I would then address memory location 0232.

Branch Instructions

There are several types of jump or branch instructions in the CHIP-8 language. Instruction 1242 would cause an unconditional branch to the instruction at memory location 0242. Instruction BMMM lets you index the branch address by adding the value of variable 0 to it before branching. Eight conditional skip instructions let you test the values of the 16 one-byte variables or determine if a specific hex key is being pressed. This latter capability is useful in video game programs. (Only the least significant hex digit of VX is used to specify the key.)

A 2570 instruction would branch to a subroutine starting at location 0570. 00EE at the end of this subroutine will return program execution to the instruction following the 2570. The subroutine itself could use another 2MMM instruction to branch to (or call) another subroutine. This technique is known as subroutine nesting. Note that all subroutines called (or branched to) by 2MMM instructions must end with 00EE. Ignoring this rule will cause hard-to-find program bugs.

How to Change and Use the Variables

The CXKK instruction sets a random byte value into VX. This random byte would have any bits matching 0 bit positions in KK set to 0. For example, a C407 instruction would set V4 equal to a random byte value between 00 and 07.

A timer (or real-time clock) can be set to any value between 00 and FF by a FX15 instruction. This timer is automatically decremented by one, 60 times per second until it reaches 00. Setting it to FF would require about 4 seconds for it to reach 00. This timer can be examined with a FX07 instruction. A FX18 instruction causes a tone to be sounded for the time specified by the value of VX. A value of FF would result in a 4-second tone. The minimum time that the speaker will respond to is that corresponding to the variable value 02.

A FX33 instruction converts the value of VX to decimal form. Suppose I=0222 and V9=A7. A F933 instruction would cause the following bytes to be stored in memory:

```
0222   01
0223   06
0224   07
```

Since A7 in hex equals 167 in decimal, we see that the three RAM bytes addressed by I contain the decimal equivalent of the value of V9.

If I=0227, a F355 instruction will cause the values of V0, V1, V2, and V3 to be stored at memory locations 0227, 0228, 0229, and 022A. If I=0210, a F265 instruction would set V0, V1, and V2 to the values of the bytes stored at RAM locations 0210, 0211, and 0212. FX55 and FX65 let you

Table I — CHIP-8 Instructions

Instruction	Operation
1MMM	Go to 0MMM
BMMM	Go to 0MMM + VO
2MMM	Do subroutine at 0MMM (must end with 00EE)
00EE	Return from subroutine
3XKK	Skip next instruction if VX = KK
4XKK	Skip next instruction if VX ≠ KK
5XY0	Skip next instruction if VX = VY
9XY0	Skip next instruction if VX ≠ VY
EX9E	Skip next instruction if VX = Hex key (LSD)
EXA1	Skip next instruction if VX ≠ Hex key (LSD)
6XKK	Let VX = KK
CXKK	Let VX = Random Byte (KK = Mask)
7XKK	Let VX = VX + KK
8XY0	Let VX = VY
8XY1	Let VX = VX/VY (VF changed)
8XY2	Let VX = VX & VY (VF changed)
8XY4	Let VX = VX + VY (VF = 00 if VX + VY ≤ FF, VF = 01 if VX + VY > FF)
8XY5	Let VX = VX − VY (VF = 00 if VX < VY, VF = 01 if VX ≥ VY)
FX07	Let VX = current timer value
FX0A	Let VX = hex key digit (waits for any key pressed)
FX15	Set timer = VX (01 = 1/60 second)
FX18	Set tone duration = VX (01 = 1/60 second)
AMMM	Let I = 0MMM
FX1E	Let I = I + VX
FX29	Let I = 5-byte display pattern for LSD of VX
FX33	Let MI = 3-decimal digit equivalent of VX (I unchanged)
FX55	Let MI = VO : VX (I = I + X + 1)
FX65	Let VO : VX = MI (I = I + X + 1)
00E0	Erase display (all 0's)
DXYN	Show n-byte MI pattern at VX-VY coordinates. I unchanged. MI pattern is combined with existing display via EXCLUSIVE-OR function. VF = 01 if a 1 in MI pattern matches 1 in existing display.
0MMM	Do machine language subroutine at 0MMM (subroutine must end with D4 byte)

store the values of variables in RAM and set the values of variables to RAM bytes. A sequence of variables (V0 to VX) is always transferred to or from RAM. If X=0, only V0 is transferred.

The 8XY1, 8XY2, and 8XY4, and 8XY5 instructions perform logic and binary arithmetic operations on two 1-byte variables. VF is used for overflow in the arithmetic operations.

Using the Display Instructions

An 00E0 instruction erases the screen to all 0's. When the CHIP-8 language is used, 256 bytes of RAM are displayed on the screen as an array of spots 64 wide by 32 high. A white spot represents a 1 bit in RAM, while a dark (or off) spot represents a 0 bit in RAM. Each spot position on the screen can be located by a pair of coordinates as shown in Fig. 1.

The VX byte value specifies the number of horizontal spot positions from the upper left corner of the display. The VY byte value specifies the number of vertical spot positions from the upper left corner of the display.

The DXYN instruction is used to show a pattern of spots on the screen. Suppose we wanted to form the pattern for the digit "8" on the screen. First we make up a pattern of bits to form "8" as shown in Fig. 2.

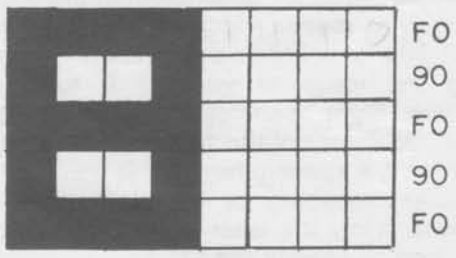


Fig. 2 — Pattern of bits forming digit 8.

In this example we made the "8" pattern five spots high by four spots wide. Patterns to be shown on the screen using the DXYN instruction must always be one byte wide and no more than

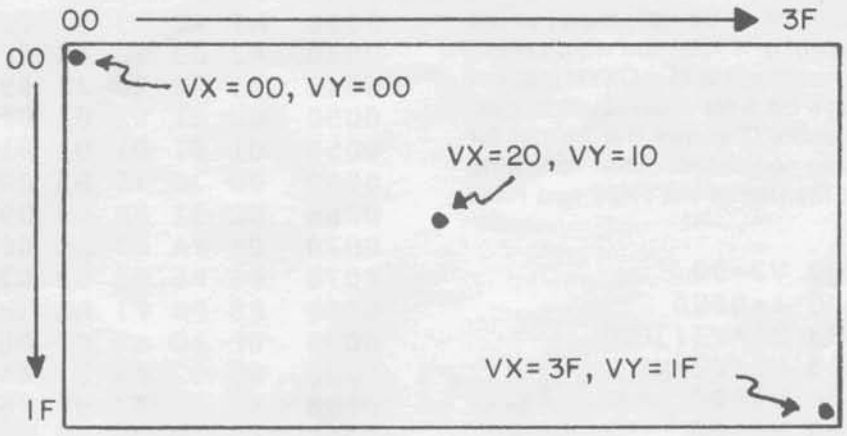


Fig. 1 — Display screen coordinate structure.

fifteen bytes high. (Several small patterns can be combined to form larger ones on the screen when required.) To the right of the "8" pattern in Fig. 2 are the equivalent byte values in hex form. We could now store this pattern as a list of five bytes at RAM location 020A as follows:

```
020A  F0
020B  90
020C  F0
020D  90
020E  F0
```

Suppose we now want to show this pattern in the upper left corner of the screen. We'll assign V1=VX and V2=VY. Now we let V1=V2=00 and set I=020A. If we now do a D125 instruction, the "8" pattern will be shown on the screen in the upper left corner.

You can write a program to show the "8" pattern on the screen as follows:

```
0200 A20A I=020A
0202 6100 V1=00
0204 6200 V2=00
0206 D125 SHOW 5MI@V1V2
0208 1208 GO 0208
020A F090
020C F090
020E F000
```

The first column of this program shows the memory locations at which the instruction bytes in the second column are stored. The third column indicates the function performed by each instruction in shorthand form. Only the bytes in the second column are actually stored in memory.

With the CHIP-8 interpreter stored at 0000-01FF, you can load the above program in memory and run it. Set V1 and V2 to different values to relocate the "8" pattern on the screen. The VX-VY coordinates always specify the screen position of the upper left-hand bit of your pattern. This bit can be either 0 or 1. The last digit of the DXYN instruction specifies the height of your patterns or the number of bytes in your pattern list.

When a pattern is displayed, it is compared with any pattern already on the screen. If a 1 bit in your pattern matches a 1 bit already on the screen, then a 0 bit will be shown at this spot position and VF will be set to a value of 01. You can test VF following a DXYN instruction to determine if your pattern touched any part of a previously displayed pattern. This feature permits programming video games which require knowing if one moving pattern touches or hits another pattern.

Because trying to display two 1 spots at the same position on the screen results in a 0 spot, you can use the DXYN instruction to erase a previously displayed pattern by displaying it a second time in the same position. (The entire screen can be erased with a single 00E0 instruction.) The following program shows the "8" pattern, shows it again to erase it, and then changes VX and VY coordinates to create a moving pattern:

```
0200 A210 I=0210
0202 6100 V1=00
0204 6200 V2=00
0206 D125 SHOW 5MI@V1V2
0208 D125 SHOW 5MI@V1V2
020A 7101 V1+01
020C 7201 V2+01
020E 1206 GO 0206
0210 F090
0212 F090
0214 F000
```

The "8" pattern byte list was moved to 0210 to make room for the other instructions. Try changing the values that V1 and V2 are incremented by for different movement speeds and angles. A delay could be inserted between the two DXYN instructions for slower motion.

The FX29 instruction sets I to the RAM address of a five-byte pattern representing the least

significant hex digit of VX. If VX=07, then I would be set to the address of a "7" pattern which could then be shown on the screen with a DXYN instruction. N should always be 5 for these built-in hex-digit patterns. Appendix C shows the format for these standard hex patterns. The following program illustrates the use of the FX29 and FX33 instructions:

```
0200 6300 V3=00
0202 A300 I=0300
0204 F333 MI=V3(3DD)
0206 F265 V0:V2=MI
0208 6400 V4=00
020A 6500 V5=00
020C F029 I=V0(LSDP)
020E D455 SHOW 5MI@V4V5
0210 7405 V4+05
0212 F129 I=V1(LSDP)
0214 D455 SHOW 5MI@V4V5
0216 7405 V4+05
0218 F229 I=V2(LSDP)
021A D455 SHOW 5MI@V4V5
021C 6603 V6=03
021E F618 TONE=V6
0220 6620 V6=20
0222 F615 TIME=V6
0224 F607 V6=TIME
0226 3600 SKIP;V6 EQ 00
0228 1224 GO 0224
022A 7301 V3+01
022C 00E0 ERASE
022E 1202 GO 0202
```

This program continuously increments V3, converts it to decimal form, and displays it on the screen.

The FX0A instruction waits for a hex key to be pressed, VX is then set to the value of the pressed key, and program execution continues when the key is released. (If key 3 is pressed, VX=03.) A tone is heard while the key is pressed. This instruction is used to wait for keyboard input.

CHIP-8 Interpreter Listing

To use the CHIP-8 language you must first load the following interpreter program into memory locations 0000-01FF (2 pages). This interpreter will allow you to run the games in Appendix D or write your own programs using the CHIP-8 instruction set described in section III.

```
0000  91 BB FF 01 B2 B6 F8 CF
0008  A2 F8 81 B1 F8 46 A1 90
0010  B4 F8 1B A4 F8 01 B5 F8
0018  FC A5 D4 96 B7 E2 94 BC
0020  45 AF F6 F6 F6 F6 32 44
0028  F9 50 AC 8F FA 0F F9 F0
0030  A6 05 F6 F6 F6 F6 F9 F0
0038  A7 4C B3 8C FC 0F AC 0C
0040  A3 D3 30 1B 8F FA 0F B3
0048  45 30 40 22 69 12 D4 00
0050  00 01 01 01 01 01 01 01
0058  01 01 01 01 01 00 01 01
0060  00 7C 75 83 8B 95 B4 B7
0068  BC 91 EB A4 D9 70 99 05
0070  06 FA 07 BE 06 FA 3F F6
0078  F6 F6 22 52 07 FA 1F FE
0080  FE FE F1 AC 9B BC 45 FA
0088  0F AD A7 F8 D0 A6 93 AF
0090  87 32 F3 27 4A BD 9E AE
0098  8E 32 A4 9D F6 BD 8F 76
00A0  AF 2E 30 98 9D 56 16 8F
00A8  56 16 30 8E 00 EC F8 D0
00B0  A6 93 A7 8D 32 D9 06 F2
00B8  2D 32 BE F8 01 A7 46 F3
00C0  5C 02 FB 07 32 D2 1C 06
00C8  F2 32 CE F8 01 A7 06 F3
00D0  5C 2C 16 8C FC 08 AC 3B
00D8  B3 F8 FF A6 87 56 12 D4
00E0  9B BF F8 FF AF 93 5F 8F
00E8  32 DF 2F 30 E5 00 42 B5
00F0  42 A5 D4 8D A7 87 32 AC
00F8  2A 27 30 F5 00 00 00 00
0100  00 00 00 00 00 45 A3 98
0108  56 D4 F8 81 BC F8 95 AC
0110  22 DC 12 56 D4 06 B8 D4
0118  06 A8 D4 64 0A 01 E6 8A
0120  F4 AA 3B 28 9A FC 01 BA
0128  D4 F8 81 BA 06 FA 0F AA
0130  0A AA D4 E6 06 BF 93 BE
0138  F8 1B AE 2A 1A F8 00 5A
0140  0E F5 3B 4B 56 0A FC 01
0148  5A 30 40 4E F6 3B 3C 9F
0150  56 2A 2A D4 00 22 86 52
0158  F8 F0 A7 07 5A 87 F3 17
0160  1A 3A 5B 12 D4 22 86 52
0168  F8 F0 A7 0A 57 87 F3 17
0170  1A 3A 6B 12 D4 15 85 22
0178  73 95 52 25 45 A5 86 FA
0180  0F B5 D4 45 E6 F3 3A 82
0188  15 15 D4 45 E6 F3 3A 88
0190  D4 45 07 30 8C 45 07 30
0198  84 E6 62 26 45 A3 36 88
01A0  D4 3E 88 D4 F8 F0 A7 E7
01A8  45 F4 A5 86 FA 0F 3B B2
01B0  FC 01 B5 D4 45 56 D4 45
01B8  E6 F4 56 D4 45 FA 0F 3A
01C0  C4 07 56 D4 AF 22 F8 D3
01C8  73 8F F9 F0 52 E6 07 D2
01D0  56 F8 FF A6 F8 00 7E 56
01D8  D4 19 89 AE 93 BE 99 EE
01E0  F4 56 76 E6 F4 B9 56 45
01E8  F2 56 D4 45 AA 86 FA 0F
01F0  BA D4 00 00 00 00 00 00
01F8  00 00 00 00 00 E0 00 4B
```

CHIP-8 Memory Map

Location	Use
0000 • • • 01FF	CHIP-8 LANGUAGE INTERPRETER
0200 • • •	User programs using CHIP-8 instruction set (1184 bytes available in 2048-byte system)
0YA0 • • • 0YCF	CHIP-8 stack (48 bytes max. for up to 12 levels of subroutine nesting)
0YD0 • • • 0YEF	Reserved for CHIP-8 INTERPRETER work area
0YF0	V0
0YF1	V1
0YF2	V2
0YF3	V3
0YF4	V4
0YF5	V5
0YF6	V6
0YF7	V7
0YF8	V8
0YF9	V9
0YFA	VA
0YFB	VB
0YFC	VC
0YFD	VD
0YFE	VE
0YFF	VF
0X00 • • • 0XFF	256-byte RAM area for display refresh

OX = Highest on-card RAM page (03 for 1024-byte system)
OY = OX − (02 for 1024-byte system)

*The CHIP-8 interpreter uses all 16 registers when executing CHIP-8 instructions. Therefore, these registers cannot be used to store machine language subroutine data while CHIP-8 instructions are being executed.

CDP1802 Register Use for CHIP-8 Interpreter

R0 = DMA pointer (page 0X for display refresh)
R1 = INTERRUPT routine program counter
R2 = Stack pointer
R3 = INTERPRETER subroutine program counter
R4 = CALL subroutine program counter
R5 = CHIP-8 instruction program counter
R6 = VX pointer (R6.1 must not be changed)
R7 = VY pointer (available for machine-language subroutines)
R8 = Timers (R8.1 = timer, R8.0 = tone duration)
R9 = Random number (+1 in INTERRUPT routine)
RA = I pointer
RB = Display page pointer (RB.1 = 0X)
RC = Available*
RD = Available*
RE = Available*
RF = Available*

CHIP-8 User Notes

1. Do not use any of the CDP1802 three-cycle machine language instructions in CHIP-8 programs.
2. CDP1802 R5 is used as the CHIP-8 instruction counter. It will be addressing the byte following a 0MMM instruction for machine language subroutines and can be used to pass 2-byte parameters. Refer to the operating system register table in Chapter 6 to examine this register during CHIP-8 program debugging.
3. Display page 0X is erased to all 0's before beginning CHIP-8 programs at 0200. To inhibit erasing page 0X, change 00E0 at location 01FC to 11FE.
4. To change the display page from 0X, use a machine language subroutine to set RB.1 equal to the new display page.
5. R7, RC, RD, RE, and RF can be used as working registers in machine language subroutines. Changing other registers can cause the CHIP-8 interpreter to malfunction.
6. Do not turn off the CDP1861 video display chip in machine language subroutines. This will interfere with proper operation of the CHIP-8 interpreter.
7. Program bugs can destroy the CHIP-8 interpreter at locations 0000-01FF. If you suspect that this has happened, reload the interpreter.
8. The CHIP-8 interpreter uses subroutines and digit patterns contained in the operating system ROM. If you modify this operating system, the CHIP-8 interpreter should not be used.

IV. Machine Language Programming

VIP Machine Coding

A listing of CDP1802 machine language instructions is provided in this chapter. For a complete description of machine language instructions, refer to the User Manual for the CDP1802 COSMAC Microprocessor MPM-201B. Your COSMAC VIP computer incorporates the following special machine-language input and output instructions:

CODE	OPERATION
69	Turn display on (Bus → MX,D)
6B	Input port byte → MX,D (Optional)
61	Turn display off (MX → Bus,RX+1)
62	MX(LSD) → Hex keyboard latch, RX+1
63	MX → Output port, RX+1 (Optional)
64	MX → Bus,RX+1

One 64 instruction is always executed by the Operating System. It can also be used in expanded systems if desired. Instructions 65, 66, 67, 6A, 6C, 6D, 6E, and 6F are also available for use in expanded systems.

The External Flag lines are used as follows:

FLAG	USE
EF1	Generated by the video interface (CDP1861)
EF2	Serial data from cassette player
EF3	Hex key pressed signal
EF4	Not used in basic system

EF4 can be used for system expansion. EF3 can also be used in expanded systems if no key will be depressed at the same time that an external device is using EF3. EF1 can only be used by an external device when the display is turned off. EF2 should not be used in expanded systems.

The latched Q line output performs several functions in the COSMAC VIP system. When set, it holds the Q light on and generates a continuous speaker tone. The Q line is also used for serial output data to a cassette recorder. You can use the Q output line as a control signal in an expanded system if you avoid conflicts with its normal functions.

You can store a machine language program starting at location 0000. It will be executed when you flip the RUN switch up. Initially P=0, X=0, R0=0000, Q=0, and R1=0XFF, where 0X = last page of on-card RAM (0X=03 in 1024-byte RAM system). The operating system uses the last 84 bytes of on-card RAM. You should avoid using these last 84 RAM bytes when writing machine language programs. With a 1024-byte RAM, locations 03AC-03FF would be reserved for use by the operating system. Note that R1 initially contains the address of the last on-card RAM byte. Your machine language program can use R1 to determine the amount of RAM in your system when required.

Putting Machine Coding and CHIP-8 Language Together

The operating system and the CHIP-8 language interpreter use a video display format that is 64 bits wide by 32 bits high. This 256-byte display can easily be modified by writing your own video refresh interrupt routine as explained in the CDP1861 data sheet provided in the VP-711 Manual (VIP-311). Display formats up to 64 bits wide by 128 bits high are possible with no hardware modification. The 4096 bit picture program in Chapter VII uses a machine language refresh routine that provides a format 64 bits wide by 64 bits high.

The CHIP-8 language described in the previous section, permits machine language subroutines to be called with a 0MMM instruction. A D4 machine language instruction at the end of the machine

language subroutine returns control to the CHIP-8 instruction following the 0MMM instruction. In the CDP1802 register use for the CHIP-8 language is provided. R5 is used as the CHIP-8 program counter. When you call a machine language subroutine with a 0MMM instruction, R5 will be addressing the CHIP-8 instruction following the 0MMM. The machine language subroutine could retrieve the next two CHIP-8 program bytes as parameters by addressing with R5 and incrementing it by 2 before returning control to the CHIP-8 program with a D4 instruction. RC, RD, RE, and RF are available for use in machine language subroutines. RA is the CHIP-8 memory pointer (I). Changing the high-order byte of RB will cause any desired RAM page to be displayed. R3 is the machine language subroutine program counter.

CHIP-8 uses the operating system refresh interrupt routine contained in ROM for display. You can use this ROM interrupt routine for 256-byte display in your own machine language programs. First initialize R1 to 8146 and R2 as a stack pointer before turning on the video interface with a 69 instruction. Set the desired display page into RB.1. This interrupt routine uses R0 as the display refresh pointer and modifies RB.0. R8.1 and R8.0 are decremented by 1 during each interrupt unless they are equal to 00. Interrupts occur 60 times per second when the video interface is turned on. This rate is controlled by a crystal clock so that R8.0 and R8.1 can be used as real-time clocks when needed.

While the video interface is turned on, you should not use any of the 3-machine-cycle CDP1802 instructions (except those used for sync in the refresh interrupt routine itself). If you are not using the video interface, then you can use the CDP1802 3-cycle instructions in your machine language programs. When you initiate a machine language program at 0000 by flipping RUN up, the video interface will be off. You must turn it on with a 69 instruction to use the COSMAC VIP graphic display capability.

Machine Language Programming Summed Up

In summary, COSMAC VIP provides you with an easy-to-use language called CHIP-8. You can insert machine language subroutines in CHIP-8 programs for greater flexibility or expanded I/O capability. You can write complete machine language programs to fully utilize CDP1802 capabilities. The operating system facilitates debugging machine language programs by permitting you to examine general registers R3-RF. (See Chapter VI.) Advanced programmers can even develop their own interpretive language tailored to special requirements. Direct execution of machine language code starting at location 0000 together with the expansion interface permits the COSMAC VIP system to be used as a low-cost development system as well as a recreational or educational computer.

CDP1802 Instruction Summary

The COSMAC instruction summary is given in Tables I and II. Hexadecimal notation is used to refer to the 4-bit binary codes.

In all registers bits are numbered from the least significant bit (LSB) to the most significant bit (MSB) starting with 0.

R(W): Register designated by W, where W=N or X, or P
R(W).0: Lower-order byte of R(W)
R(W).1: Higher-order byte of R(W)
N0 = Least significant Bit of N Register

Operation Notation
M(R(N)) → D; R(N) + 1

This notation means: The memory byte pointed to by R(N) is loaded into D, and R(N) is incremented by 1.

INSTRUCTION SUMMARY
by Class of Operation

Register Operations

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
INCREMENT REG N	INC	1N	R(N) +1
DECREMENT REG N	DEC	2N	R(N) −1
INCREMENT REG X	IRX	60	R(X) +1
GET LOW REG N	GLO	8N	R(N).0→D
PUT LOW REG N	PLO	AN	D→R(N).0
GET HIGH REG N	GHI	9N	R(N).1→D
PUT HIGH REG N	PHI	BN	D→R(N).1

Memory Reference

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
LOAD VIA N	LDN	0N	M(R(N))→D; FOR N NOT 0
LOAD ADVANCE	LDA	4N	M(R(N))→D;R(N) +1
LOAD VIA X	LDX	F0	M(R(X))→D
LOAD VIA X AND ADVANCE	LDXA	72	M(R(X))→D; R(X) +1
LOAD IMMEDIATE	LDI	F8	M(R(P))→D; R(P) + 1
STORE VIA N	STR	5N	D→M(R(N))
STORE VIA X AND DECREMENT	STXD	73	D→M(R(X)); R(X) −1

Logic Operations♦♦

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
OR	OR	F1	M(R(X)) OR D→D
OR IMMEDIATE	ORI	F9	M(R(P)) OR D→D; R(P) +1
EXCLUSIVE OR	XOR	F3	M(R(X)) XOR D→D
EXCLUSIVE OR IMMEDIATE	XRI	FB	M(R(P)) XOR D→D; R(P) +1
AND	AND	F2	M(R(X)) AND D→D
AND IMMEDIATE	ANI	FA	M(R(P)) AND D→D; R(P) +1
SHIFT RIGHT	SHR	F6	SHIFT D RIGHT, LSB(D)→DF, 0→MSB(D)
SHIFT RIGHT WITH CARRY	SHRC	76♦	SHIFT D RIGHT, LSB(D)→DF, DF→MSB(D)
RING SHIFT RIGHT	RSHR		
SHIFT LEFT	SHL	FE	SHIFT D LEFT, MSB(D)→DF, 0→LSB(D)
SHIFT LEFT WITH CARRY	SHLC	7E♦	SHIFT D LEFT, MSB(D)→DF, DF→LSB(D)
RING SHIFT LEFT	RSHL		

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.
♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF.

Arithmetic Operations♦♦

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
ADD	ADD	F4	M(R(X)) +D→DF, D
ADD IMMEDIATE	ADI	FC	M(R(P)) +D→DF, D; R(P) +1
ADD WITH CARRY	ADC	74	M(R(X)) +D +DF→DF, D
ADD WITH CARRY IMMEDIATE	ADCI	7C	M(R(P)) +D +DF→DF, D R(P) +1
SUBTRACT D	SD	F5	M(R(X))−D→DF, D
SUBTRACT D IMMEDIATE	SDI	FD	M(R(P))−D→DF, D; R(P) +1
SUBTRACT D WITH BORROW	SDB	75	M(R(X))−D−(NOT DF)→DF, D
SUBTRACT D WITH BORROW, IMMEDIATE	SDBI	7D	M(R(P))−D−(NOT DF)→DF, D; R(P) +1
SUBTRACT MEMORY	SM	F7	D−M(R(X))→DF, D
SUBTRACT MEMORY IMMEDIATE	SMI	FF	D−M(R(P))→DF, D; R(P) +1
SUBTRACT MEMORY WITH BORROW	SMB	77	D−M(R(X))−(NOT DF)→DF, D
SUBTRACT MEMORY WITH BORROW, IMMEDIATE	SMBI	7F	D−M(R(P))−(NOT DF)→DF, D R(P) +1

Branch Instructions — Short Branch

SHORT BRANCH	BR	30	M(R(P))→R(P).0
NO SHORT BRANCH (SEE SKP)	NBR	38♦	R(P) +1
SHORT BRANCH IF D=0	BZ	32	IF D=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF D NOT 0	BNZ	3A	IF D NOT 0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF DF=1	BDF	33♦	IF DF=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF POS OR ZERO	BPZ		
SHORT BRANCH IF EQUAL OR GREATER	BGE		
SHORT BRANCH IF DF=0	BNF	3B♦	IF DF=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF MINUS	BM		
SHORT BRANCH IF LESS	BL		
SHORT BRANCH IF Q=1	BQ	31	IF Q=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF Q=0	BNQ	39	IF Q=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF1=1 (1 = V_{SS})	B1	34	IF EF1=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF1=0 (0 = V_{CC})	BN1	3C	IF EF1=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF2=1 (1 = V_{SS})	B2	35	IF EF2=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF2=0 (0 = V_{CC})	BN2	3D	IF EF2=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF3=1 (1 = V_{SS})	B3	36	IF EF3=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF3=0 (0 = V_{CC})	BN3	3E	IF EF3=0, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF4=1 (1 = V_{SS})	B4	37	IF EF4=1, M(R(P))→R(P).0 ELSE R(P) +1
SHORT BRANCH IF EF4=0 (0 = V_{CC})	BN4	3F	IF EF4=0, M(R(P))→R(P).0 ELSE R(P) +1

Control Instructions

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
IDLE	IDL	00	WAIT FOR DMA OR INTERRUPT; M(R(0))→BUS
NO OPERATION	NOP	C4	CONTINUE
SET P	SEP	DN	N→P
SET X	SEX	EN	N→X
SET Q	SEQ	7B	1→Q
RESET Q	REQ	7A	0→Q
SAVE	SAV	78	T→M(R(X))
PUSH X,P TO STACK	MARK	79	(X,P)→T; (X,P)→M(R(2)) THEN P→X; R(2)−1
RETURN	RET	70	M(R(X))→(X,P); R(X) +1 1→IE
DISABLE	DIS	71	M(R(X))→(X,P); R(X) +1 0→IE

Input–Output Byte Transfer

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
OUTPUT 1	OUT 1	61	M(R(X))→BUS; R(X) +1; N LINES = 1
OUTPUT 2	OUT 2	62	M(R(X))→BUS; R(X) +1; N LINES = 2
OUTPUT 3	OUT 3	63	M(R(X))→BUS; R(X) +1; N LINES = 3
OUTPUT 4	OUT 4	64	M(R(X))→BUS; R(X) +1; N LINES = 4
OUTPUT 5	OUT 5	65	M(R(X))→BUS; R(X) +1; N LINES = 5
OUTPUT 6	OUT 6	66	M(R(X))→BUS; R(X) +1; N LINES = 6
OUTPUT 7	OUT 7	67	M(R(X))→BUS; R(X) +1; N LINES = 7
INPUT 1	INP 1	69	BUS→M(R(X)); BUS→D; N LINES = 1
INPUT 2	INP 2	6A	BUS→M(R(X)); BUS→D; N LINES = 2
INPUT 3	INP 3	6B	BUS→M(R(X)); BUS→D; N LINES = 3
INPUT 4	INP 4	6C	BUS→M(R(X)); BUS→D; N LINES = 4
INPUT 5	INP 5	6D	BUS→M(R(X)); BUS→D; N LINES = 5
INPUT 6	INP 6	6E	BUS→M(R(X)); BUS→D; N LINES = 6
INPUT 7	INP 7	6F	BUS→M(R(X)); BUS→D; N LINES = 7

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.
♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF

Branch Instructions — Long Branch

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
LONG BRANCH	LBR	C0	M(R(P))→R(P).1 M(R(P) +1)→R(P).0
NO LONG BRANCH (SEE LSKP)	NLBR	C8♦	R(P) +2
LONG BRANCH IF D=0	LBZ	C2	IF D=0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF D NOT 0	LBNZ	CA	IF D NOT 0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF DF=1	LBDF	C3	IF DF=1, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF DF=0	LBNF	C8	IF DF=0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF Q=1	LBQ	C1	IF Q=1, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2
LONG BRANCH IF Q=0	LBNQ	C9	IF Q=0, M(R(P))→R(P).1 M(R(P) +1)→R(P).0 ELSE R(P) +2

Skip Instructions

INSTRUCTION	MNEMONIC	OP CODE	OPERATION
SHORT SKIP (SEE NBR)	SKP	38♦	R(P) +1
LONG SKIP (SEE NLBR)	LSKP	C8♦	R(P) +2
LONG SKIP IF D=0	LSZ	CE	IF D=0, R(P) +2 ELSE CONTINUE
LONG SKIP IF D NOT 0	LSNZ	C6	IF D NOT 0, R(P) +2 ELSE CONTINUE
LONG SKIP IF DF=1	LSDF	CF	IF DF=1, R(P) +2 ELSE CONTINUE
LONG SKIP IF DF=0	LSNF	C7	IF DF=0, R(P) +2 ELSE CONTINUE
LONG SKIP IF Q=1	LSQ	CD	IF Q=1, R(P) +2 ELSE CONTINUE
LONG SKIP IF Q=0	LSNQ	C5	IF Q=0, R(P) +2 ELSE CONTINUE
LONG SKIP IF IE=1	LSIE	CC	IF IE=1, R(P) +2 ELSE CONTINUE

♦NOTE: THIS INSTRUCTION IS ASSOCIATED WITH MORE THAN ONE MNEMONIC. EACH MNEMONIC IS INDIVIDUALLY LISTED.
♦♦NOTE: THE ARITHMETIC OPERATIONS AND THE SHIFT INSTRUCTIONS ARE THE ONLY INSTRUCTIONS THAT CAN ALTER THE DF

V. Logic Description

This system is designed around the CDP1802 microprocessor (U1). Refer to the CDP1802 data sheet and User Manual for the CDP1802 COSMAC Microprocessor MPM-201B for a complete description of its operation. The CDP1802 requires a square-wave clock input at pin 1 for operation. This system uses a 1.7609-MHz clock. One half of U3 is connected as a free-running crystal-controlled oscillator. A 3.52180-MHz crystal is used in this circuit. The output of this 3.52180-MHz oscillator is then divided by 2 using U4 to provide the 1.7609-MHz input clock for the CDP1802. Because each CDP1802 machine cycle equals 8 clock cycles, each machine cycle is about 4.54 μs in duration. TPA and TPB are timing pulses generated once each machine cycle by the CDP1802 microprocessor.

How Memory Is Addressed

A debounced RUN level goes high when the RUN switch is flipped up. This signal causes the CDP1802 to begin fetching instructions from memory. When the RUN switch is down, the CDP1802 is held in a reset state and U6A (in Fig. 2) is reset. U6B is held set by U6A. The CDP1802 starts fetching instructions from the ROM (U10) at location 8000 since U6B is being held set. The ROM contains the operating system program which uses a 64 instruction to generate an N2 pulse. This N2 pulse sets U6A so it no longer holds U6B in its set state. From this point on, the selection of RAM or ROM locations is controlled by the most significant address bit latched into U6B each cycle by TPA.

U8 latches an additional 4 address bits to provide the 12-bit address required in a 4096-byte RAM system. U9A decodes 2 of these address bits into 4 lines which are used to select up to four 1024-byte RAM sections. Each 1024-byte section of RAM consists of two 4 x 1024-bit RAM IC's (U16-U23 in Fig. 4). Only the first section of RAM (U16-U17) are used in a 1024-byte system. U9B in Fig. 2 is wired as a simple gate that inhibits selecting any section of RAM when either the ROM is selected or a positive RAM inhibit signal is generated on pin 19 of the expansion interface by external circuits.

Memory read ($\overline{\text{MRD}}$) and write ($\overline{\text{MWR}}$) signals are supplied to the RAM at appropriate times by the CDP1802. Data is transferred between memory, CDP1802, input, or output via an 8-bit data bus. Pull-up resistors are provided on this bus for compatibility with TTL signal swings provided by some RAMs.

How the Input/Output Works

The VP-111 can be expanded to include an input/output port.

U11 and U12 in Fig. 3 are used to decode the input/output instruction codes used in the system.

U13 provides the hex keyboard interface. This interface permits a program to determine which key is pressed. A 62 machine instruction causes the least significant 4 bits of memory byte to be latched into U13. These 4 bits are decoded to bring one of the 16 U13 output lines low. If the key that corresponds to this output line is pressed, the CDP1802 EF3 input will go low. The 4-bit codes latched into U13 correspond to the equivalent key position. After the program sends a 4-bit code to U13, it subsequently examines the EF3 line to see if the key corresponding to this code is pressed or not. In this manner, a program can determine when any specific key is pressed or can sequentially scan all keys while waiting for any one to be pressed. Key debounce delays must be provided

in the program when required. A program can also cause a speaker tone to occur when a key is pressed. Only one key at a time should be pressed with this method of interfacing the keyboard.

U15 generates an audible tone when pin 4 is high. The output on pin 3 drives a small speaker. The 10-ohm resistor R48 in series with the speaker output can be raised in value to lower the volume if desired. The CDP1802 latched Q-line output drives the tone generator and also turns on the Q light. Q can be set high (1) or low (0) by machine language instructions. The RC network connected to pins 2, 6, and 7 of U15 determines the frequency of the tone. You can increase or decrease the value of R to adjust this frequency to suit your taste.

Q is also shaped by U14A in Fig. 3 to form a signal suitable for recording on an audio cassette. Audio cassette recorders can't cope with square waves. The divider on the output of U14A reduces the signal to about 50 mV which is suitable for the microphone input of most recorders. During recording, the operating system program in ROM converts memory bytes into bit serial form and transmits them to the recorder via the Q line.

In playback, bit serial data from the cassette drives the tape light. The serial data is amplified and shaped into 5-volt pulses by U14B. The output of U14B is connected to the CDP1802 EF2 input line. The operating system reads tape data by examining the timing of the transitions on the EF2 input line. Cassette read and record timing is derived from the crystal-controlled clock so that no adjustments are necessary.

Video output is provided by the unique CDP1861 video display interface IC (U2 in Fig. 1). This chip provides one of the lowest cost and most useful display interface capabilities available for any microcomputer. The 61 and 69 machine language instructions are used to generate the required on and off pulses for U2. The down position of the RUN switch resets the internal U2 circuits. When a program is initiated, by flipping RUN up, U2 will remain off until a 69 instruction is executed. No CDP1802 interrupt or DMA requests are generated by U2 until it is turned on by a 69 instruction. U1 and U2 are both driven by the same clock. They must remain in sync to provide proper operation of the display.

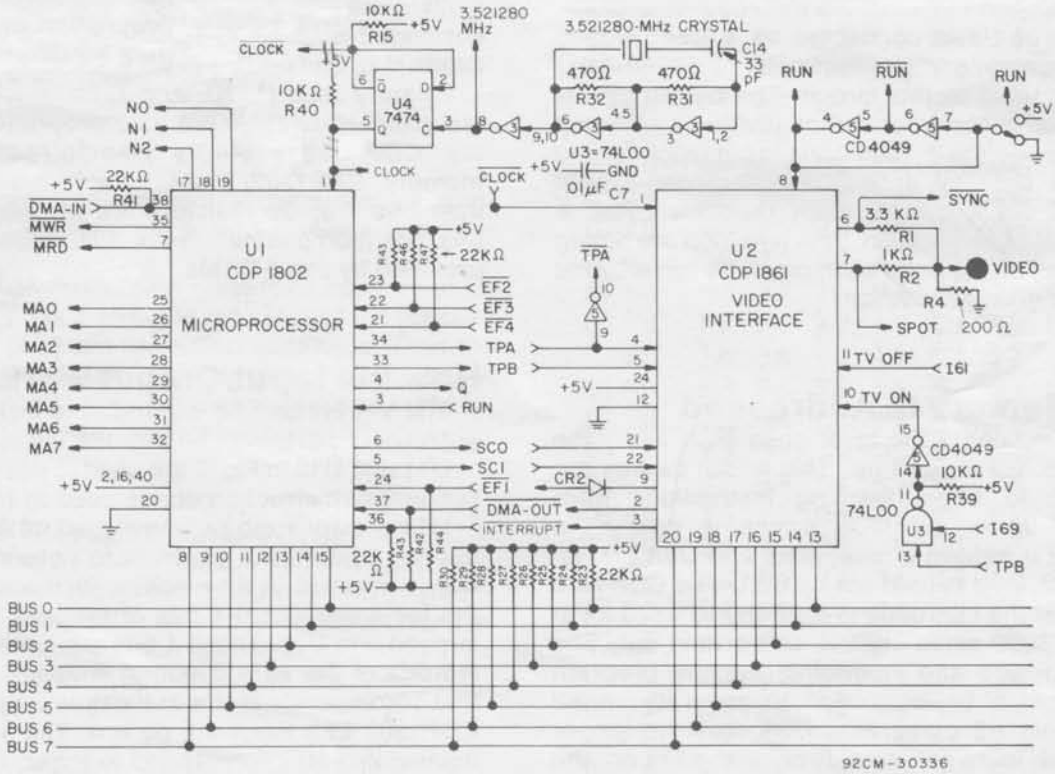


Fig. 1 — Microprocessor and Display Interface Circuits

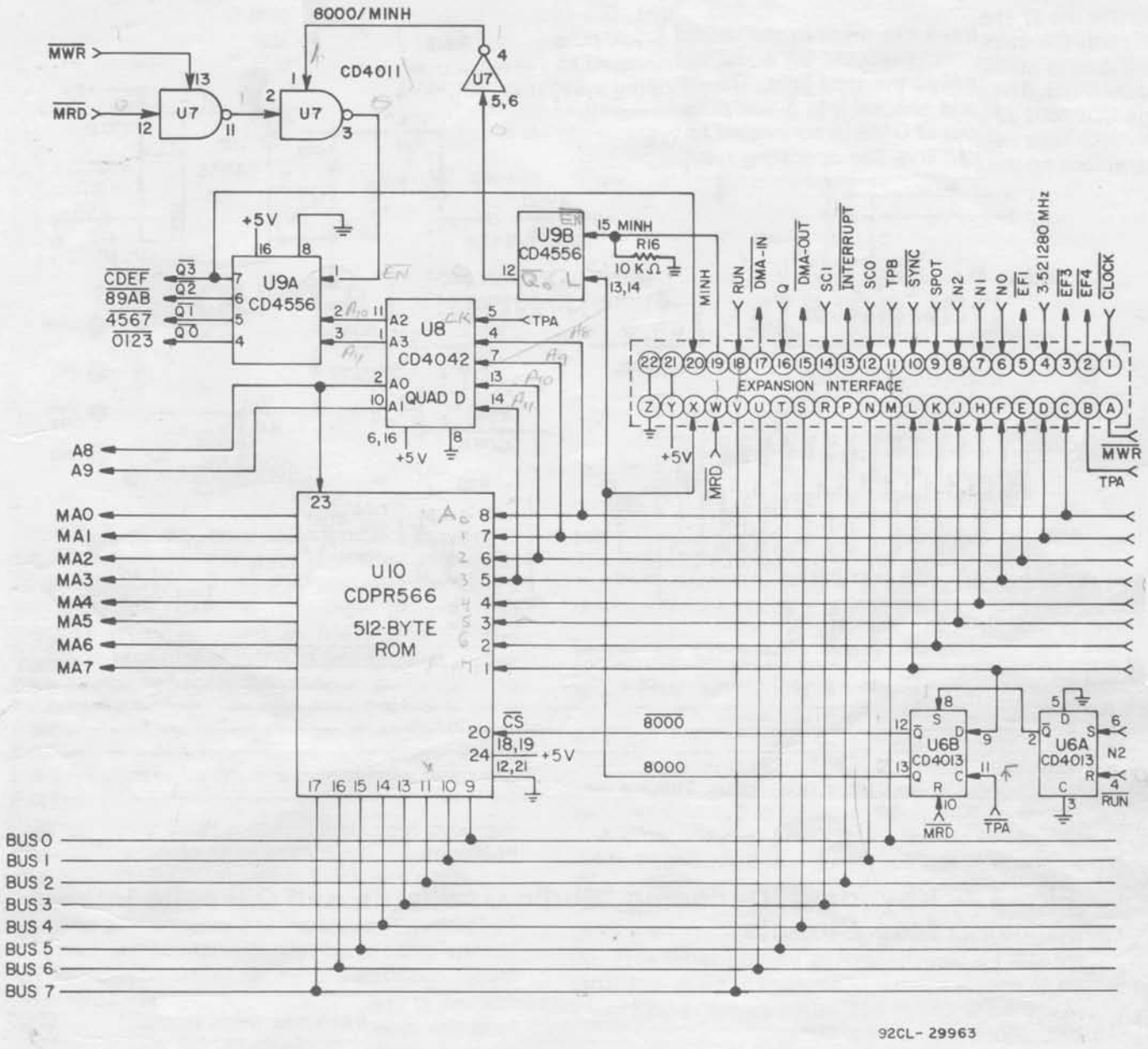


Fig. 2 — ROM Circuits and Expansion Interface

Note: Expansion interface connector provided with VP-114 Expansion Kit for the VP-111.

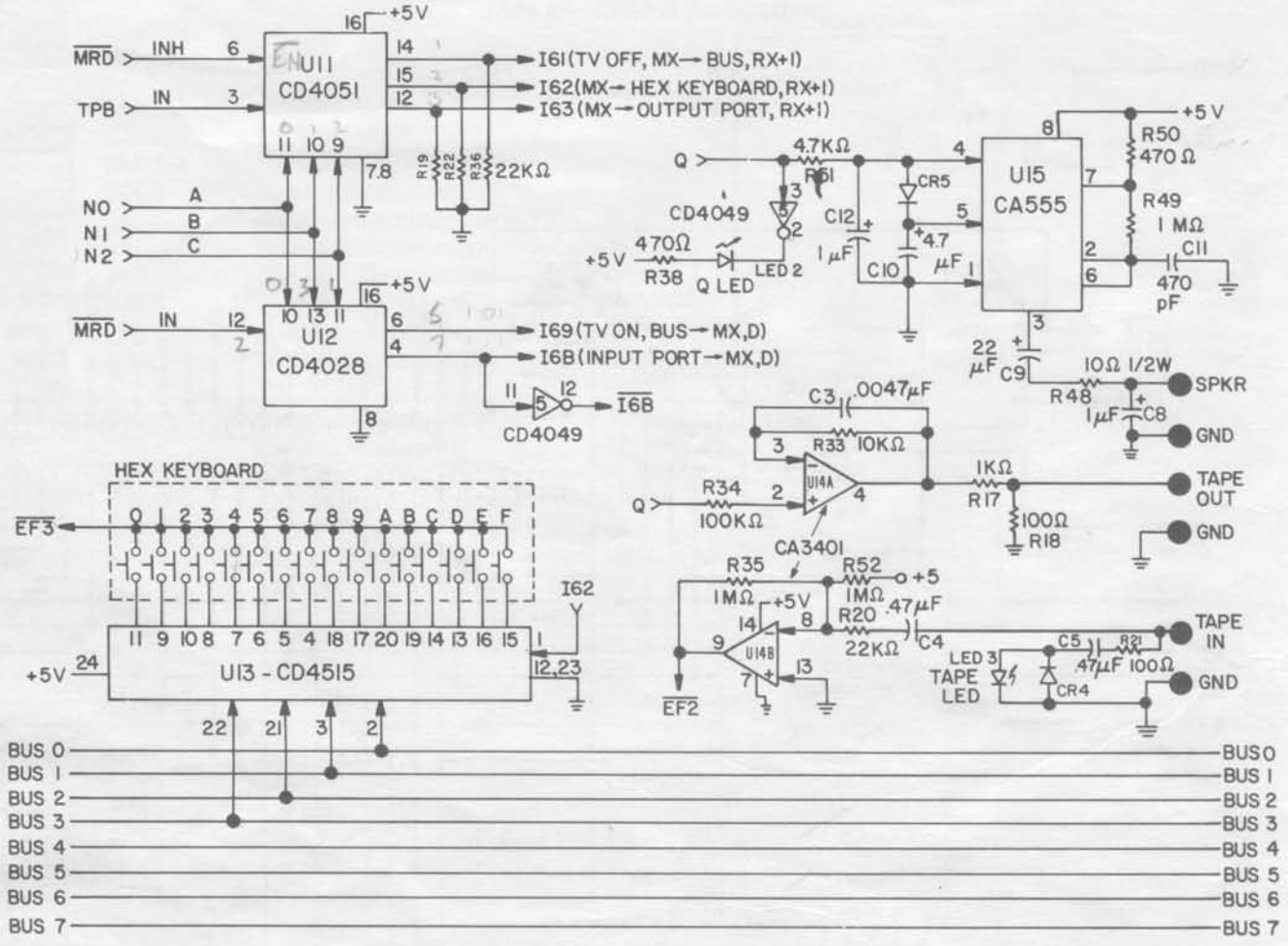


Fig. 3 — Keyboard, Decoding, Audio Oscillator, and Cassette Interface Circuits

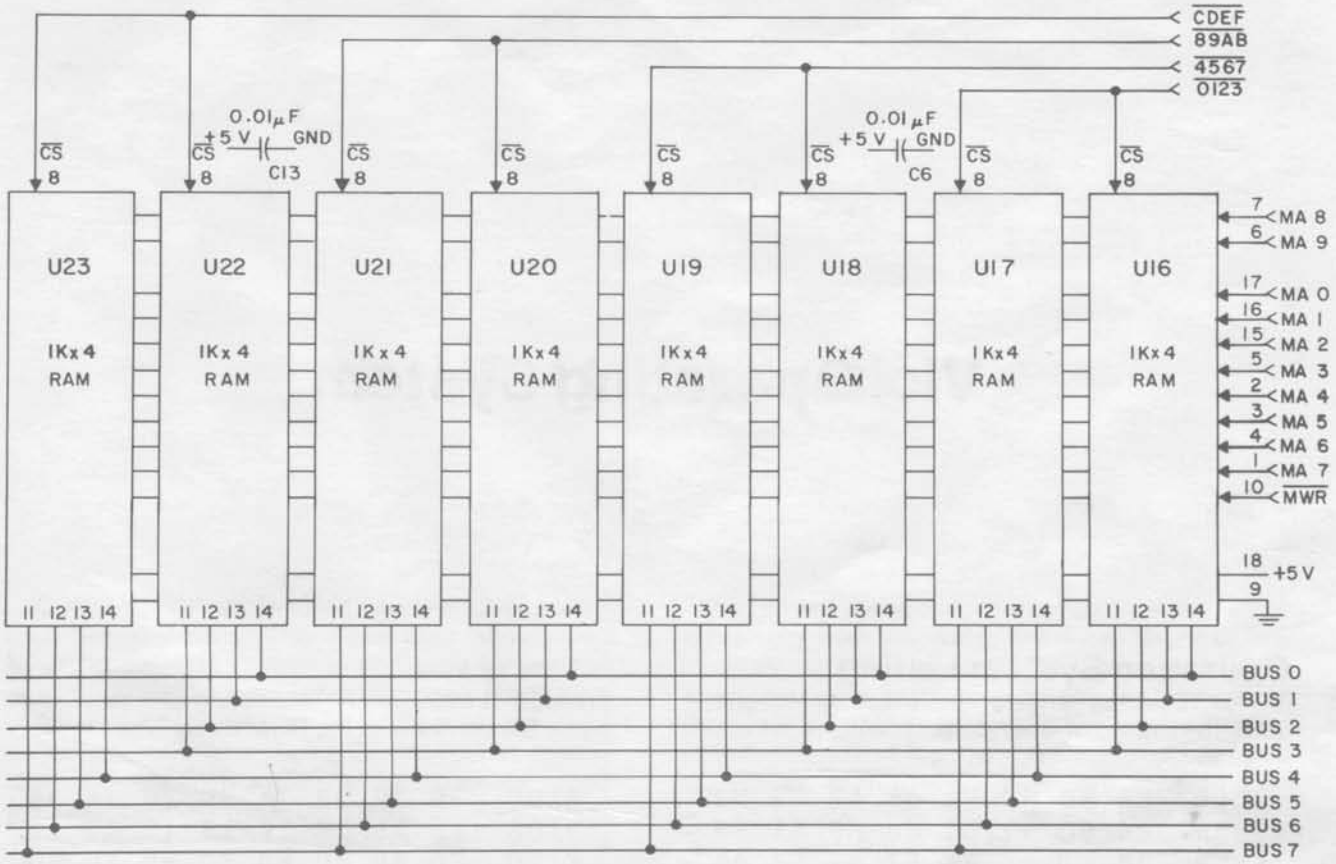
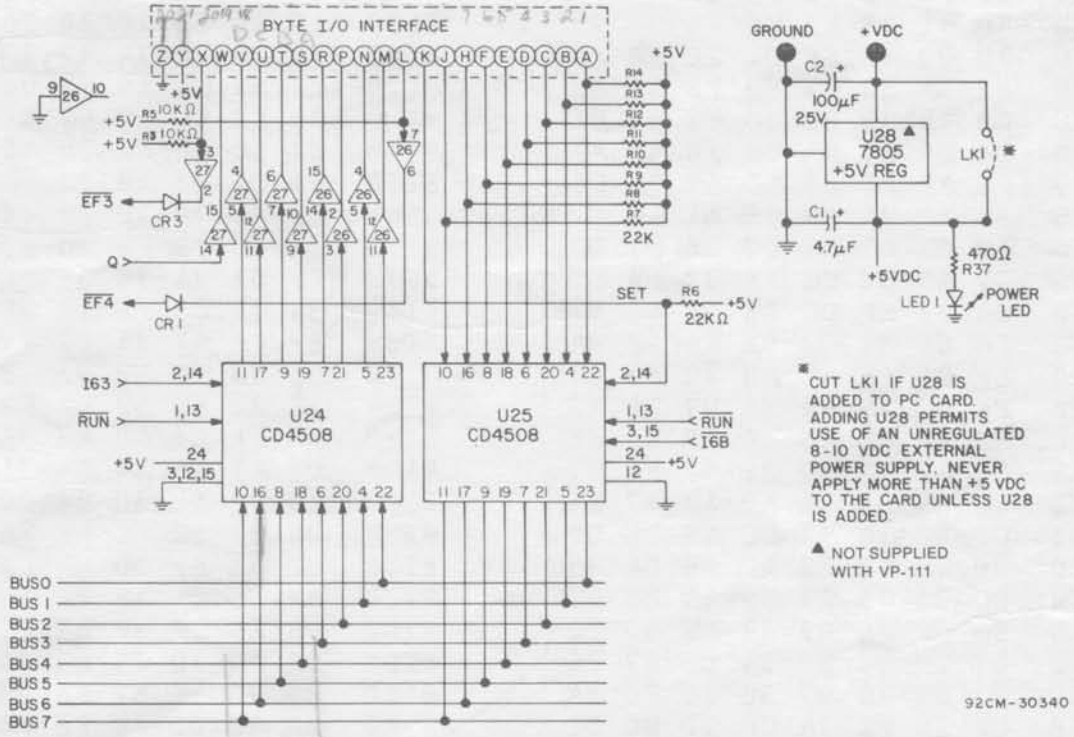


Fig. 4 — RAM Circuits

92CM-30329

Note: RAM's, U18 through U23, plus sockets for these RAM's are provided with VP-114 Expansion Kit for the VP-111.



92CM-30340

Fig. 5 — Power Supply Circuit and Byte Input/Output Interface

NOTE: This schematic provided for reference only. Parts to implement the input/output port are provided with VP-114 Expansion Kit for the VP-111.

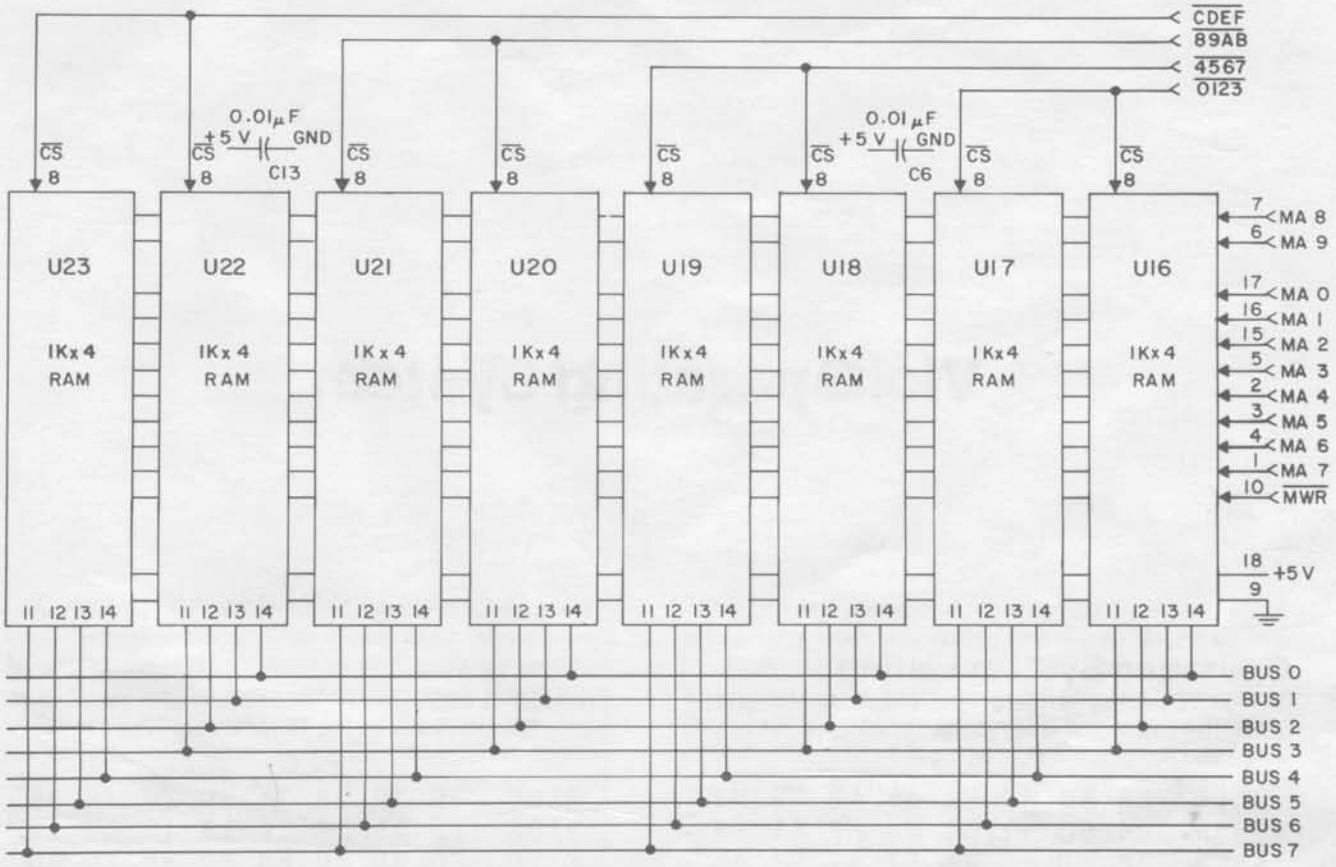
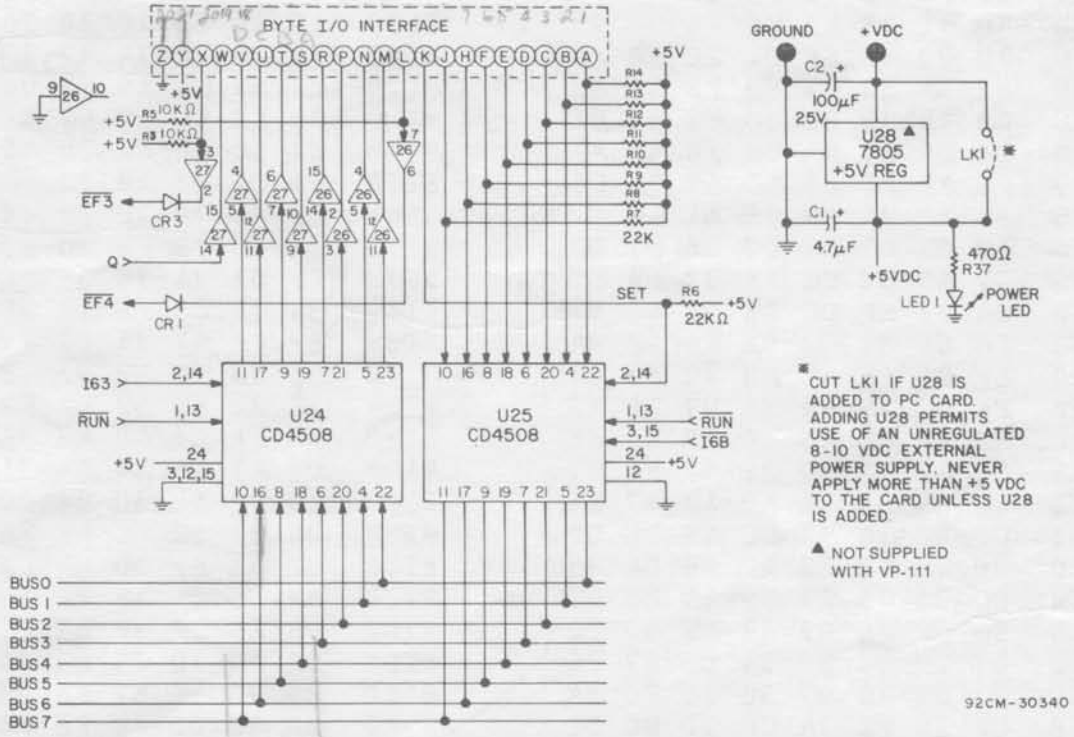


Fig. 4 — RAM Circuits

92CM-30329

Note: RAM's, U18 through U23, plus sockets for these RAM's are provided with VP-114 Expansion Kit for the VP-111.



92CM-30340

Fig. 5 — Power Supply Circuit and Byte Input/Output Interface

NOTE: This schematic provided for reference only. Parts to implement the input/output port are provided with VP-114 Expansion Kit for the VP-111.

Operating System Register Table

Memory Address	Register Byte	Memory Address	Register Byte
0XB0	—	0XC0	—
0XB1	—	0XC1	—
0XB2	—	0XC2	—
0XB3	R3.0	0XC3	R3.1
0XB4	R4.0	0XC4	R4.1
0XB5	R5.0	0XC5	R5.1
0XB6	R6.0	0XC6	R6.1
0XB7	R7.0	0XC7	R7.1
0XB8	R8.0	0XC8	R8.1
0XB9	R9.0	0XC9	R9.1
0XBA	RA.0	0XCA	RA.1
0XBB	RB.0	0XCB	RB.1
0XBC	RC.0	0XCC	RC.1
0XBD	RD.0	0XCD	RD.1
0XBE	RE.0	0XCE	RE.1
0XBF	RF.0	0XCF	RF.1

0X = 03 for 1024-byte RAM
0X = 07 for 2048-byte RAM
0X = 0B for 3072-byte RAM
0X = 0F for 4096-byte RAM
R5 = CHIP-8 language program counter
RA = CHIP-8 language I pointer

Operating System Summary

1. RUN up with key C pressed selects operating system at 8000.
2. Enter four-digit address followed by mode digit:

A = MR (Memory Read)
0 = MW (Memory Write)
B = TR (Tape Read)
F = TW (Tape Write)

3. CDP1802 microprocessor registers are stored as shown in table above. They may be examined after a program is run by using operating system mode A.
4. Mode 0 can be used to insert temporary stops in a program for debugging purposes. Insert a "branch-to-itself" instruction at the desired stopping point.
5. The operating system uses the top 84 bytes of RAM (0XAC-0XFF). Avoid using these byte locations in your programs.
6. The operating system searches for and uses the top (highest) 256-byte page of on-card RAM. When RUN is flipped up to execute a program beginning at 0000, the following initial conditions exist:

P=0, Q=0, R0=0000, and R1=0XFF where 0X = highest page of on-card RAM.

VII. Video Games

1. VIP Kaleidoscope

This program uses the CHIP-8 INTERPRETER at 0000-01FF. Four spots appear in a group at the center of the screen. Press keys 2, 4, 6, or 8 to create a pattern. Keep your pattern smaller than 138 key depressions. Push key 0 to terminate pattern entry. Pushing key 0 causes your pattern to be continuously repeated forming a fascinating, changing kaleidoscope display on the screen. A "44444442220" key sequence provides a very nice effect. Experiment to find other nice patterns. The subroutine at 0232-0274 causes your pattern to be duplicated in the four quadrants of the screen.

```
0200 6000 V0=00
0202 6380 V3=80
0204 611F V1=1F
0206 620F V2=0F
0208 2232 DO 0232
020A A200 I=0200
020C F31E I=I+V3
020E F00A V0=KEY
0210 F055 MI=V0:V0
0212 4000 SKIP;V0 NE 00
0214 121C GO 021C
0216 7301 V3+01
0218 3300 SKIP;V3 EQ 00
021A 1208 GO 0208
021C 6380 V3=80
021E A200 I=0200
0220 F31E I=I+V3
0222 F065 V0:V0=MI
0224 4000 SKIP;V0 NE 00
0226 121C GO 021C
0228 7301 V3+01
022A 4300 SKIP;V3 NE 00
022C 121C GO 021C
022E 2232 DO 0232
0230 121E GO 021E
0232 4002 SKIP;V0 NE 02
0234 72FF V2+FF
0236 4004 SKIP;V0 NE 04
0238 71FF V1+FF
023A 4006 SKIP;V0 NE 06
023C 7101 V1+01
023E 4008 SKIP;V0 NE 08
0240 7201 V2+01
0242 A277 I=0277
0244 6AE0 VA=E0
0246 8A12 VA=VA&V1
0248 6B1F VB=1F
024A 81B2 V1=V1&VB
024C 3A00 SKIP;VA EQ 00
024E 7201 V2+01
0250 6AF0 VA=F0
0252 8A22 VA=VA&V2
0254 6B0F VB=0F
0256 82B2 V2=V2&VB
0258 3A00 SKIP;VA EQ 00
025A 7101 V1+01
025C 6B1F VB=1F
025E 81B2 V1=V1&VB
0260 D121 SHOW 1MI@V1V2
0262 8A10 VA=V1
0264 6B1F VB=1F
0266 8B25 VB=VB-V2
0268 DAB1 SHOW 1MI@VAVB
026A 6A3F VA=3F
026C 8A15 VA=VA-V1
026E DAB1 SHOW 1MI@VAVB
0270 8B20 VB=V2
0272 DAB1 SHOW 1MI@VAVB
0274 00EE RET
0276 0180
0278 0000
```

2. VIP Video Display Drawing Game

This program uses the CHIP-8 INTERPRETER at 0000-01FF. A flashing spot appears in the upper left corner of the screen. You can move the spot by holding key 2, 4, 6, or 8. Press key 5 and you can draw a picture with the spot. Press key 0 and the spot can be moved without drawing or used to erase a previously drawn line. 0245-024E is a list of initial values for V0-V9. In this program, locations 0300-03FF are used for the picture. After drawing a picture, you can change M(0208) from 00E0 to 120A. Write locations 0000-03FF (4 pages) to tape to save your picture. When you load these four pages back into memory you will see your original picture. Changing the 00E0 instruction in the program to 120A prevents your picture from being erased when the program is started.

```
0200 A245 I=0245
0202 F965 V0:V9=MI
0204 A24F I=024F
0206 0236 MLS@0236
0208 00E0 ERASE
020A F915 TIME=V9
020C FA07 VA=TIME
020E 3A00 SKIP;VA EQ 00
0210 120C GO 020C
0212 D121 SHOW 1MI@V1V2
0214 3F00 SKIP;VF EQ 00
0216 D121 SHOW 1MI@V1V2
0218 E3A1 SKIP;V3 NE KEY
021A 8030 V0=V3
021C E4A1 SKIP;V4 NE KEY
021E 8040 V0=V4
0220 4000 SKIP;V0 NE 00
0222 123C GO 023C
0224 E5A1 SKIP;V5 NE KEY
0226 72FF V2+FF
0228 E6A1 SKIP;V6 NE KEY
022A 71FF V1+FF
022C E7A1 SKIP;V7 NE KEY
022E 7101 V1+01
0230 E8A1 SKIP;V8 NE KEY
0232 7201 V2+01
0234 120A GO 020A
0236 01F8
0238 03BB
023A E2D4
023C D121 SHOW 1MI@V1V2
023E 4F00 SKIP;VF NE 00
0240 D121 SHOW 1MI@V1V2
0242 1224 GO 0224
0244 0100
0246 0000
0248 0005
024A 0204
024C 0608
024E 0880
```

3. VIP Spooky Spot

This program uses the CHIP-8 INTERPRETER at locations 0000-01FF. Now you can let the computer make your big decisions or predict the future just like government or industry leaders do. Flip RUN up. You will see the words YES and NO at the right of the screen. Ask the computer any question that can be answered with YES or NO. Press KEY 0 and the spooky spot will show you the computer's answer. This program replaces your old fashioned mechanical OUIJA board.

```
0200 00E0 ERASE
0202 2242 DO 0242
0204 2254 DO 0254
0206 FA0A VA=KEY
0208 A290 I=0290
020A 6100 V1=00
020C 6210 V2=10
020E D121 SHOW 1MI@V1V2
0210 3F00 SKIP;VF EQ 00
0212 1236 GO 0236
0214 6A04 VA=04
0216 FA18 TONE=VA
0218 6A0A VA=0A
021A FA15 TIME=VA
021C FA07 VA=TIME
021E 3A00 SKIP;VA EQ 00
0220 121C GO 021C
0222 7101 V1+01
0224 CA01 VA=RND
0226 3A01 SKIP;VA EQ 01
0228 6AFF VA=FF
022A 82A4 V2=V2+VA
022C 4207 SKIP;V2 NE 07
022E 7201 V2+01
0230 4218 SKIP;V2 NE 18
0232 72FF V2+FF
0234 120E GO 020E
0236 6A10 VA=10
0238 8A22 VA=VA&V2
023A 3A00 SKIP;VA EQ 00
023C 1240 GO 0240
023E 225A DO 025A
0240 226A DO 026A
0242 A270 I=0270
0244 6408 V4=08
0246 6330 V3=30
0248 D348 SHOW 8MI@V3V4
024A 6A08 VA=08
024C FA1E I=I+VA
024E 7308 V3+08
0250 D348 SHOW 8MI@V3V4
0252 00EE RET
0254 A280 I=0280
0256 6410 V4=10
0258 1246 GO 0246
025A 6408 V4=08
025C 6331 V3=31
025E A290 I=0290
0260 D348 SHOW 8MI@V3V4
0262 7301 V3+01
0264 3340 SKIP;V3 EQ 40
0266 1260 GO 0260
0268 1268 GO 0268
026A 6410 V4=10
026C 125C GO 025C
026E 0101
0270 7F7F
0272 6A6A
0274 6276
0276 767F
0278 FFFF
027A 23EF
027C 63FB
027E 23FF
0280 7F76
0282 7270
0284 7476
0286 7F7F
0288 FF87
028A B7B7
028C B787
028E FFFF
0290 8080
0292 8080
0294 8080
0296 8080
0298 80D4
```

4. VIP Hi-Lo

This program uses the CHIP-8 INTERPRETER at 0000-01FF. You have 10 chances to guess the value of a random number between 00 and 99 selected by the program. The number at the right of the screen shows the number of the guess you are using. Enter a two digit number and the computer tells you if you are high or low. Press any key to erase this number and then, try again. If you have failed after ten guesses, press any key and the number will be shown. If you are good you will never need more than seven guesses. If you are not so good, alter the program to allow more guesses by changing location 0292 from 4E0A to 4E99.

```
0200 6C09 VC=09
0202 CD0F VD=RND
0204 8CD5 VC=VC-VD
0206 4F00 SKIP;VF NE 00
0208 1200 GO 0200
020A 89D0 V9=VD
020C 6C09 VC=09
020E CD0F VD=RND
0210 8CD5 VC=VC-VD
0212 4F00 SKIP;VF NE 00
0214 120C GO 020C
0216 8AD0 VA=VD
0218 6E00 VE=00
021A A2AA I=02AA
021C 7E01 VE+01
021E FE33 MI=VE(3DD)
0220 F265 V0:V2=MI
0222 6B30 VB=30
0224 6C10 VC=10
0226 680F V8=0F
0228 F129 I=V1(LSDP)
022A DBC5 SHOW 5MI@VBVC
022C 7B05 VB+05
022E F229 I=V2(LSDP)
0230 DBC5 SHOW 5MI@VBVC
0232 4800 SKIP;V8 NE 00
0234 1254 GO 0254
0236 660A V6=0A
0238 F10A V1=KEY
023A 8165 V1=V1-V6
023C 3F00 SKIP;VF EQ 00
023E 1236 GO 0236
0240 710A V1+0A
0242 660A V6=0A
0244 F20A V2=KEY
0246 8265 V2=V2-V6
0248 3F00 SKIP;VF EQ 00
024A 1242 GO 0242
024C 720A V2+0A
024E 6B10 VB=10
0250 6800 V8=00
0252 1228 GO 0228
0254 8195 V1=V1-V9
0256 3100 SKIP;V1 EQ 00
0258 1272 GO 0272
025A 82A5 V2=V2-VA
025C 3200 SKIP;V2 EQ 00
025E 1286 GO 0286
0260 6B20 VB=20
0262 6518 V5=18
0264 F929 I=V9(LSDP)
0266 DBC5 SHOW 5MI@VBVC
0268 7B05 VB+05
026A FA29 I=VA(LSDP)
026C DBC5 SHOW 5MI@VBVC
026E FC18 TONE=VC
0270 1270 GO 0270
0272 65F0 V5=F0
0274 8152 V1=V1&V5
0276 3100 SKIP;V1 EQ 00
0278 128E GO 028E
027A A29F I=029F
027C 6B10 VB=10
027E 6C18 VC=18
0280 DBC5 SHOW 5MI@VBVC
0282 F60A V6=KEY
0284 1292 GO 0292
0286 65F0 V5=F0
0288 8252 V2=V2&V5
028A 4200 SKIP;V2 NE 00
028C 127A GO 027A
028E A2A4 I=02A4
0290 127C GO 027C
0292 4E0A SKIP;VE NE 0A
0294 129A GO 029A
0296 00E0 ERASE
0298 121A GO 021A
029A DBC5 SHOW 5MI@VBVC
029C 1260 GO 0260
029E 0197
02A0 92F2
02A2 9297
02A4 8F89
02A6 8989
02A8 EFD4
```

5. VIP 4096-Bit Picture

This is a machine language program that shows a picture pattern stored at 0100-02FF on the screen. Load the following program into memory at 0000-002F:

```
0000  90 B1 B2 B3 F8 08 A3 D3
0008  F8 FF A2 F8 14 A1 E2 69
0010  30 10 42 70 22 78 22 52
0018  C4 C4 C4 F8 01 B0 91 A0
0020  80 E2 E2 20 A0 E2 80 A0
0028  E2 20 A0 3C 20 30 12 00
```

Store the following picture pattern at 0100-02FF. This picture is shown as an array of spots 64 wide by 64 high. You can substitute your own picture pattern at 0100-01FF.

```
0100  80 00 00 00 00 80 00 00
0108  00 00 00 00 00 00 00 00
0110  00 00 00 00 00 01 FF FF
0118  00 00 00 00 00 02 00 01
0120  00 00 10 00 30 02 00 02
0128  00 00 00 00 78 01 FF FE
0130  00 00 00 3F 87 F8 20 00
0138  00 00 00 20 30 08 20 00
0140  00 00 00 3F 87 F0 20 00
0148  00 00 00 00 7A 20 20 00
0150  00 40 00 00 31 10 20 20
0158  00 00 00 00 00 C8 20 00
0160  00 00 00 00 01 3F F8 00
0168  00 00 00 00 02 00 08 00
0170  00 00 00 40 06 FC 08 00
0178  00 03 80 00 02 01 F8 00
0180  0C 07 C0 00 01 0E 00 00
0188  1E 0F E0 00 00 F0 00 00
0190  3F BF F0 00 00 00 00 00
0198  37 FF F8 00 00 00 00 00
01A0  67 FF E8 02 00 00 00 00
01A8  67 FF F8 00 00 00 00 00
01B0  43 BF F8 00 00 04 00 00
01B8  43 BF F0 00 00 00 00 00
01C0  C3 FF E0 00 00 00 00 00
01C8  C3 FF C0 00 00 00 00 00
01D0  81 E0 00 00 00 00 00 01
01D8  81 C0 00 00 00 00 00 00
01E0  81 C0 00 00 00 00 00 00
01E8  81 C0 00 00 00 00 00 00
01F0  81 C0 00 00 00 00 00 00
01F8  81 E0 00 00 00 00 00 00
0200  C1 FE 00 00 00 00 00 00
0208  C1 FE 00 00 00 00 00 00
0210  63 FE 00 00 00 00 00 00
0218  63 FC 00 00 00 00 00 00
0220  3F F8 00 00 00 00 00 00
0228  3F F0 30 00 00 00 00 00
0230  03 F0 70 EE E3 BB AB B8
0238  03 F8 F0 A8 A2 2A 3A A0
0240  00 FD E0 E8 E2 2B BB A0
0248  00 FF C0 C8 A2 28 AA A0
0250  00 7F 80 AE A3 BB AA B8
0258  00 7F 00 00 00 00 00 00
0260  00 6F 00 00 00 00 00 00
0268  00 6F 80 00 00 00 00 00
0270  00 77 80 1D D5 D5 DD C0
0278  00 77 C0 11 5D 54 91 40
0280  00 5B C0 11 5D D4 99 C0
0288  00 5B C0 11 55 14 91 80
0290  00 6D C0 1D D5 1C 9D 40
0298  00 6D C0 00 00 00 00 00
02A0  00 6D C0 00 00 00 00 00
02A8  00 6D C0 00 00 00 00 00
02B0  00 33 80 00 00 00 00 00
02B8  00 3F 3F FF FF FF FF FF
02C0  00 1F 00 00 00 00 00 00
02C8  00 0E 00 00 00 00 00 00
02D0  00 0E 00 00 00 00 00 00
02D8  00 0E 00 00 00 00 00 00
02E0  00 0E 00 00 00 00 00 00
02E8  00 1F 80 00 00 00 00 00
02F0  00 3F C0 00 00 00 00 00
02F8  00 3F C0 00 00 00 00 00
```

6. VP-111 Shooting Stars

This program uses the CHIP-8 INTERPRETER at 0000-01FF.

The large object displayed at the center of the screen is an asteroid which, when reached by the players' spacecraft, will destroy the meteor. If the spacecraft or asteroid is struck by a meteor the spacecraft is destroyed.

The movement of the spacecraft is controlled by keys 2 (up), 8 (down), 4 (left) and 6 (right). The score is incremented when meteors are destroyed and decremented when the spacecraft is destroyed.

```
0200 6000 V0=00
0202 6E00 VE=00
0204 6118 V1=18
0206 620E V2=0E
0208 A2C5 I=02C5
020A D126 SHOW 6MI@V1V2
020C C330 V3=RND
020E C41A V4=RND
0210 A2C5 I=02C5
0212 D343 SHOW 3MI@V3V4
0214 3F00 SKIP;VF EQ 00
0216 1294 GO 0294
0218 3E00 SKIP;VE EQ 00
021A 124E GO 024E
021C C13F V1=RND
021E C21F V2=RND
0220 C903 V9=RND
0222 7901 V9+01
0224 CA03 VA=RND
0226 7AFC VA+FC
0228 8B90 VB=V9
022A CC01 VC=RND
022C 3C00 SKIP;VC EQ 00
022E 1244 GO 0244
0230 CC01 VC=RND
0232 3C00 SKIP;VC EQ 00
0234 1248 GO 0248
0236 CC01 VC=RND
0238 3C00 SKIP;VC EQ 00
023A 1240 GO 0240
023C 6100 V1=00
023E 124C GO 024C
0240 6200 V2=00
0242 124C GO 024C
0244 89A0 V9=VA
0246 1230 GO 0230
0248 8AB0 VA=VB
024A 1236 GO 0236
024C 6EFF VE=FF
024E 8194 V1=V1+V9
0250 82A4 V2=V2+VA
0252 6D08 VD=08
0254 8BD4 VB=VB+VD
0256 4F00 SKIP;VF NE 00
0258 125C GO 025C
025A 6E00 VE=00
025C A270 I=0270
025E D121 SHOW 1MI@V1V2
0260 4F00 SKIP;VF NE 00
0262 1272 GO 0272
0264 70FF V0+FF
0266 A2C5 I=02C5
0268 D343 SHOW 3MI@V3V4
026A A2A0 I=02A0
026C D345 SHOW 5MI@V3V4
026E 12BE GO 02BE
0270 0200 MLS@0200
0272 A2C5 I=02C5
0274 D343 SHOW 3MI@V3V4
0276 6502 V5=02
0278 6604 V6=04
027A 6706 V7=06
027C 6808 V8=08
027E E5A1 SKIP;V5 NE KEY
0280 74FF V4+FF
0282 E6A1 SKIP;V6 NE KEY
0284 73FF V3+FF
0286 E7A1 SKIP;V7 NE KEY
0288 7301 V3+01
028A E8A1 SKIP;V8 NE KEY
028C 7401 V4+01
028E A270 I=0270
0290 D121 SHOW 1MI@V1V2
0292 1210 GO 0210
0294 A2A0 I=02A0
0296 D125 SHOW 5MI@V1V2
0298 7001 V0+01
029A F029 I=V0(LSDP)
029C D865 SHOW 5MI@V8V6
029E 12B0 GO 02B0
02A0 8850
02A2 0050
02A4 8800
02A6 0000
02A8 5500
02AA 129A
02AC 6000
02AE 129A
02B0 6540 V5=40
02B2 F515 TIME=V5
02B4 F507 V5=TIME
02B6 3500 SKIP;V5 EQ 00
02B8 12B4 GO 02B4
02BA 00E0 ERASE
02BC 1202 GO 0202
02BE 65FF V5=FF
02C0 12A8 GO 02A8
02C2 0000
02C4 003C
02C6 7EFF
02C8 FF7E
02CA 3C00
```

7. VP-111 Space Intercept

This program uses the CHIP-8 INTERPRETER at 0000-01FF. Press 1 to select the large UFO which counts 5 points when hit or 2 to select the small UFO which counts 15 points when hit. Launch your rocket by pressing key 4, 5, or 6. You get 15 rockets as shown in the lower right corner of the screen. Your score is shown in the lower left corner of the screen.

```
0200 F00A V0=KEY
0202 3002 SKIP;V0 EQ 02
0204 120A GO 020A
0206 A2BA I=02BA
0208 120C GO 020C
020A A2B4 I=02B4
020C F565 V0:V5=MI
020E A2C0 I=02C0
0210 F355 MI=V0:V3
0212 A2C0 I=02C0
0214 6600 V6=00
0216 D633 SHOW 3MI@V6V3
0218 A006 I=0006
021A 671D V7=1D
021C 681F V8=1F
021E D781 SHOW 1MI@V7V8
0220 6900 V9=00
0222 6A0F VA=0F
0224 2286 DO 0286
0226 2290 DO 0290
0228 4A00 SKIP;VA NE 00
022A 122A GO 022A
022C 671E V7=1E
022E 681C V8=1C
0230 A2B1 I=02B1
0232 D783 SHOW 3MI@V7V8
0234 6E00 VE=00
0236 6B80 VB=80
0238 6D04 VD=04
023A EDA1 SKIP;VD NE KEY
023C 6BFF VB=FF
023E 6D05 VD=05
0240 EDA1 SKIP;VD NE KEY
0242 6B00 VB=00
0244 6D06 VD=06
0246 EDA1 SKIP;VD NE KEY
0248 6B01 VB=01
024A 4B80 SKIP;VB NE 80
024C 1252 GO 0252
024E 6E01 VE=01
0250 FD18 TONE=VD
0252 A2C0 I=02C0
0254 D633 SHOW 3MI@V6V3
0256 8644 V6=V6+V4
0258 D633 SHOW 3MI@V6V3
025A 3F00 SKIP;VF EQ 00
025C 1274 GO 0274
025E 4E00 SKIP;VE NE 00
0260 1236 GO 0236
0262 A2B1 I=02B1
0264 D783 SHOW 3MI@V7V8
0266 4800 SKIP;V8 NE 00
0268 1280 GO 0280
026A 78FF V8+FF
026C 87B4 V7=V7+VB
026E D783 SHOW 3MI@V7V8
0270 3F01 SKIP;VF EQ 01
0272 1252 GO 0252
0274 FD18 TONE=VD
0276 2286 DO 0286
0278 8954 V9=V9+V5
027A 2286 DO 0286
027C A2B1 I=02B1
027E D783 SHOW 3MI@V7V8
0280 2290 DO 0290
0282 7AFF VA+FF
0284 1226 GO 0226
0286 A2C3 I=02C3
0288 F933 MI=V9(3DD)
028A 6C00 VC=00
028C 229A DO 029A
028E 00EE RET
0290 A2C3 I=02C3
0292 FA33 MI=VA(3DD)
0294 6C32 VC=32
0296 229A DO 029A
0298 00EE RET
029A 6D1B VD=1B
029C F265 V0:V2=MI
029E F029 I=V0(LSDP)
02A0 DCD5 SHOW 5MI@VCVD
02A2 7C05 VC+05
02A4 F129 I=V1(LSDP)
02A6 DCD5 SHOW 5MI@VCVD
02A8 7C05 VC+05
02AA F229 I=V2(LSDP)
02AC DCD5 SHOW 5MI@VCVD
02AE 00EE RET
02B0 0140
02B2 E0A0
02B4 7CFE
02B6 7C08
02B8 0105
02BA 60F0
02BC 6003
02BE 010F
```